Marin Ghinea
Virgiliu Fireţeanu

MATLAB

CALCUL NUMERIC
GRAFICĂ
APLICAŢII

Teora

Titlul: **MATLAB Calcul numeric. Grafică. Aplicații**

Copyright © 2003, 1997 Teora

Toate drepturile asupra acestei cărți aparțin Editurii **Teora**.
Reproducerea integrală sau parțială a textului sau a ilustrațiilor din această carte este interzisă fără acordul prealabil scris al Editurii **Teora**.

Teora
Calea Moșilor nr. 211, sector 2, București, Romania
fax: 021/210.38.28
e-mail: office@teora.ro

Teora – Cartea prin poștă
CP 79-30, cod 72450 București, Romania
tel: 021/252.14.31
e-mail: cpp@teora.ro

Copertă: Gheorghe Popescu
Director Editorial: Diana Rotaru

Președinte: Teodor Răducanu

NOT 5784 CAL MATLAB-CALCUL NUMERIC, AR
ISBN 973-601-275-1

Printed in Romania

CUPRINS

CUPRINS 5

Partea I. INTRODUCERE ÎN MATLAB

1. CE ESTE MATLAB ? 15
2. MATLAB SUB WINDOWS 20
   - 2.1. Instalarea 20
   - 2.2. Lansarea în execuţie 20
   - 2.3. Ferestrele de lucru 21
     - 2.3.1. Fereastra de comenzi 21
       - 2.3.1.1. Gestionarea fişierelor 21
       - 2.3.1.2. Editarea programelor 22
       - 2.3.1.3. Alegerea opţiunilor 23
       - 2.3.1.4. Selectarea ferestrelor 24
       - 2.3.1.5. Documentaţia de ajutor - Help 24
     - 2.3.2. Fereastra de reprezentări grafice 25
       - 2.3.2.1. Gestionarea ferestrelor grafice 25
       - 2.3.2.2. Editarea graficelor 25
3. FUNCŢII MATLAB DE INTERES GENERAL 27
   - 3.1. Funcţii pentru lansarea şi ieşirea din MATLAB 27
   - 3.2. Funcţii de control general 27
     - 3.2.1. Help-ul „on-line" 28
     - 3.2.2. Informaţii despre fişierele unui director 28
     - 3.2.3. Afişarea conţinutului unui fişier 29
     - 3.2.4. Căile de căutare 29
     - 3.2.5. Informaţii despre variabilele unui fişier 30
   - 3.3. Funcţii pentru controlul directoarelor, fişierelor şi ale sistemului de operare 30
     - 3.3.1. Controlul directoarelor şi fişierelor 31
     - 3.3.2. Rularea programelor externe 31
   - 3.4. Funcţii de control al mărimilor de timp 32
   - 3.5. Funcţii asociate ferestrei de comenzi 32
     - 3.5.1. Controlul ferestrei de comenzi 32
     - 3.5.2. Formatele de afişare a numerelor 33
     - 3.5.3. Controlul execuţiei unui program 33
   - 3.6. Funcţii de control ale variabilelor din memorie 34
     - 3.6.1. Afişarea unui text 34
     - 3.6.2. Ştergerea variabilelor 34
     - 3.6.3. Compactarea memoriei 35
     - 3.6.4. Dimensiunea variabilelor 35
   - 3.7. Variabile speciale şi constante în MATLAB 35

3.8. Importul şi exportul fişierelor de date 36

4. INTRODUCERE ÎN PROGRAMAREA MATLAB 38
4.1. Matrice, vectori şi scalari. Notaţii 38
4.2. Introducerea matricelor simple 38
4.3. Elementele matricelor 39
4.4. Declaraţii şi variabile 40
4.5. Numere şi expresii aritmetice 41
4.6. Structura unui program în MATLAB 41
4.6.1. Fişierele script 42
4.6.2. Fişierele funcţie 42
4.7. Comentariile şi help-ul într-un program 44
4.8. Un exemplu simplu de program complet 45

5. INSTRUCŢIUNI ŞI FUNCŢII DE CONTROL 47
5.1. Instrucţiuni de control logic 47
5.1.1. Instrucţiunea condiţională „if" 47
5.1.1.1. Operatorii relaţionali 47
5.1.1.2. Operatorii logici 49
5.1.1.3. Instrucţiunea „if" simplă 51
5.1.1.4. Clauza „else" 53
5.1.1.5. Clauza „elseif" 53
5.1.2. Instrucţiunea repetitivă „for" 55
5.1.3. Instrucţiunea repetitivă „while" 57
5.1.4. Instrucţiunea break 57
5.1.5. Instrucţiunea return 58
5.1.6. Instrucţiunea error 58
5.2. Funcţii de control logic 58
5.2.1. Testarea condiţiilor logice 58
5.2.2. Determinarea indicilor elementelor care verifică o condiţie 59
5.2.3. Determinarea poziţiei elementelor finite, infinite sau NaN 60
5.2.4. Testarea matricelor goale 61
5.2.5. Testarea existenţei unei variabile 61
5.3. Vectorizarea calculelor 62

6. OPERAREA CU ŞIRURI DE CARACTERE 63
6.1. Funcţii generale 63
6.2. Compararea şirurilor de caractere 66
6.3. Conversia şir-număr şi număr-şir 67
6.4. Conversia şirurilor în formate numerice 69

Partea a II-a. CALCUL NUMERIC CU MATLAB

7. OPERAŢII ARITMETICE 71
7.1. Operaţii aritmetice cu scalari 71
7.1.1. Ordinea operaţiilor aritmetice 72
7.1.2. Limitele calculelor 72
7.2. Operaţii aritmetice cu tablouri 73
7.2.1. Adunarea şi scăderea 73

7.2.2. Înmulţirea tablourilor 73
7.2.3. Împărţirea la dreapta 74
7.2.4. Împărţirea la stânga 74
7.2.5. Ridicarea la putere 75
7.2.6. Transpunerea tablourilor 75
7.3. Operaţii aritmetice cu vectori 76
7.3.1. Produsul scalar 76
7.3.2. Produsul vectorial 76
7.4. Operaţii aritmetice cu matrice 78
7.4.1. Adunarea şi scăderea 78
7.4.2. Înmulţirea matricelor 79
7.4.3. Împărţirea la dreapta 79
7.4.4. Împărţirea la stânga 80
7.4.5. Ridicarea la putere 80
7.4.6. Transpunerea matricelor 81

8. GENERAREA VECTORILOR ŞI A MATRICELOR UZUALE 82
8.1. Generarea vectorilor 82
8.1.1. Generarea vectorilor cu pas liniar 82
8.1.2. Generarea vectorilor cu pas logaritmic 83
8.2. Generarea matricelor 84
8.2.1. Matricea goală 84
8.2.2. Matricea unitate 84
8.2.3. Matricea zero 85
8.2.4. Matricea identitate 85
8.2.5. Matricea aleatoare 86
8.3. Generarea unei reţele (mesh) 87

9. FUNCŢII MATEMATICE UZUALE 89
9.1. Aproximarea numerelor 89
9.1.1. Aproximarea cu numere întregi 89
9.1.2. Aproximarea cu numere raţionale 90
9.1.3. Aproximarea cu fracţii continue 91
9.1.4. Funcţia rest 92
9.1.5. Funcţia semn 92
9.2. Divizori şi multipli comuni 93
9.2.1. Cel mai mare divizor comun 93
9.2.2. Cel mai mic multiplu comun 93
9.3. Numere complexe 93
9.3.1. Definirea numerelor complexe în MATLAB 94
9.3.2. Modulul şi argumentul numerelor complexe 94
9.3.3. Părţile reală şi imaginară şi conjugatul numerelor complexe 95
9.4. Funcţiile putere, radical, logaritm şi exponenţială 96
9.4.1. Funcţia putere 97
9.4.2. Funcţia radical 98
9.4.3. Funcţia logaritm 98
9.4.4. Funcţia exponenţială 99

9.5. Funcţiile trigonometrice 99
9.5.1. Funcţiile trigonometrice directe 99
9.5.2. Funcţiile trigonometrice inverse 100
9.6. Funcţiile hiperbolice 101
9.6.1. Funcţiile hiperbolice directe 101
9.6.2. Funcţiile hiperbolice inverse 101

10. MATRICE SPECIALE 102
10. 1. Companionul matriceal 102
10. 2. Matricea diagonală 103
10. 3. Matricea Hadamard 104
10. 4. Matricea Hankel 105
10. 5. Matricea Hilbert 106
10. 6. Inversa matricei Hilbert 106
10. 7. Tensorul Kroneker 107
10. 8. Pătratul magic 107
10. 9. Matricea Pascal 108
10.10. Matricea Toeplitz 109
10.11. Matricea Vandermonde 110
10.12. Matricea Wilkinson 111

11. CALCULE CU MATRICE 112
11.1. Manipularea matricelor 112
11.1.1. Extragerea submatricelor prin indici 112
11.1.2. Extragerea submatricelor prin vectori cu elemente 0 şi 1 115
11.1.3. Asamblarea matricelor mari 115
11.1.4. Redimensionarea unei matrice 115
11.1.5. Rotirea matricei în jurul unei coloane sau linii 116
11.1.6. Rotirea matricei cu multipli de 90 de grade în jurul unui element 117
11.1.7. Matrice diagonale şi triunghiulare 117
11.2. Analiză matriceală 118
11.2.1. Determinantul unei matrice 118
11.2.2. Inversa unei matrice 119
11.2.3. Rangul unei matrice 119
11.2.4. Urma unei matrice 120
11.2.5. Calculul normelor vectorilor şi matricelor 120
11.2.6. Condiţionarea unei matrice 121
11.3. Descompunerea şi factorizarea matricelor 123
11.3.1. Valori şi vectori proprii 123
11.3.2. Calculul valorilor singulare 124
11.3.3. Factorizarea Cholesky 125
11.3.4. Factorizarea LU (lower-upper) 126
11.3.5. Factorizarea QR 127
11.3.6. Pseudoinversa unei matrice 127

12. PRELUCRAREA DATELOR ŞI CALCULE STATISTICE 129
12.1. Maximul şi minimul 129

12.2. Media şi mediana 130
12.3. Sume şi produse 130
12.4. Sortarea elementelor unei matrice 132
12.5. Varianţa şi dispersia datelor 133
12.6. Coeficientul de corelaţie a datelor 134
12.7. Valori nedefinite 135
12.8. Eliminarea datelor eronate 135
12.9. Histograme 136

13. MINIMIZAREA FUNCŢIILOR 137
13.1. Calculul minimului funcţiilor de o variabilă 137
13.2. Calculul minimului funcţiilor de mai multe variabile 138
13.3. Calculul zerourilor funcţiilor de o variabilă reală 139

14. REZOLVAREA SISTEMELOR DE ECUAŢII LINIARE 140
14.1. Descrierea matriceală a sistemelor de ecuaţii 140
14.2. Rezolvarea sistemelor prin împărţirea matricelor 140
14.3. Rezolvarea sistemelor prin folosirea matricei inverse 141

15. CALCULE NUMERICE CU POLINOAME 143
15.1. Evaluarea polinoamelor 143
15.2. Adunarea şi scăderea 144
15.3. Înmulţirea şi împărţirea 145
15.4. Descompunerea în fracţii simple 146
15.5. Calculul derivatei 147
15.6. Calculul rădăcinilor 148

16. INTERPOLAREA ŞI APROXIMAREA DATELOR 149
16.1. Căutarea datelor în tabele 149
16.1.1. Căutarea datelor în tabele unidimensionale 149
16.1.2. Căutarea datelor în tabele bidimensionale 150
16.2. Interpolarea funcţiilor de o singură variabilă 151
16.2.1. Interpolarea liniară 152
16.2.2. Interpolarea spline cubică 153
16.2.3. Interpolarea prin metoda transformatei Fourier 154
16.2.4. Interpolarea multiplă 155
16.3. Interpolarea funcţiilor de două variabile 156
16.3.1. Interpolarea după o lege biarmonică 156
16.3.2. Interpolarea după o lege biliniară 156
16.3.3. Interpolarea după o lege bicubică 157
16.3.4. Interpolarea prin metoda distanţei inverse 158
16.3.5. Interpolarea multiplă 159
16.4. Aproximarea prin metoda celor mai mici pătrate 159
16.4.1. Regresia liniară 159
16.4.2. Regresia polinomială 161

17. INTEGRAREA ŞI DERIVAREA NUMERICĂ A FUNCŢIILOR 164
17.1. Integrarea numerică 164
17.1.1. Metoda trapezelor şi metoda lui Simpson 164

17.1.2. Calculul numeric al integralelor pe intervale infinite 165
17.1.3. Calculul numeric al integralelor funcţiilor cu singularităţi 166
17.1.4. Funcţii MATLAB pentru integrarea numerică 166
17.2. Derivarea numerică 167
17.2.1. Aproximarea derivatelor cu diferenţe finite 168
17.2.2. Aproximarea numerică a derivatelor 168
17.2.3. Derivarea numerică bazată pe polinomul de interpolare Newton 171
17.2.4. Aproximarea numerică a Laplaceanului 171
17.2.5. Aproximarea numerică a gradientului 171

18. INTEGRAREA NUMERICĂ A ECUAŢIILOR DIFERENŢIALE 174
18.1. Ecuaţii diferenţiale de ordinul întâi 174
18.2. Funcţiile MATLAB pentru integrarea numerică a ecuaţiilor diferenţiale 175
18.3. Integrarea ecuaţiilor diferenţiale de ordin superior şi a sistemelor de ecuaţii diferenţiale 177

19. FUNCŢII SPECIALE 182
19.1. Ecuaţia şi funcţiile Bessel 182
19.2. Funcţia Gamma 185
19.3. Funcţia Beta 185
19.4. Funcţia eroare 186
19.5. Funcţia de repartiţie Laplace 187

Partea a III-a. GRAFICĂ ÎN MATLAB

20. REPREZENTĂRI GRAFICE 2D 189
20.1. Reprezentări grafice elementare 189
20.1.1. Reprezentarea grafică în coordonate liniare 189
20.1.2. Reprezentarea grafică în coordonate logaritmice şi semilogaritmice 191
20.1.3. Reprezentarea grafică a poligoanelor 192
20.2. Reprezentări grafice speciale X-Y 192
20.2.1. Reprezentarea grafică în coordonate polare 193
20.2.2. Reprezentarea graficelor cu bare 193
20.2.3. Reprezentarea discretă a datelor 194
20.2.4. Reprezentarea graficelor în trepte 195
20.2.5. Reprezentarea grafică a erorilor 196
20.2.6. Reprezentarea grafică a histogramelor 196
20.2.7. Reprezentarea grafică a vectorilor 198
20.2.8. Reprezentări grafice cu parametri impuşi 199
20.2.9. Reprezentări grafice dinamice 2D 200

21. REPREZENTĂRI GRAFICE 3D 201
21.1. Reprezentarea liniilor de contur 201
21.1.1. Calculul matricei liniilor de contur 201
21.1.2. Reprezentarea grafică în plan a liniilor de contur 201
21.1.3. Etichetarea cotelor liniilor de contur 203

21.1.4. Reprezentarea grafică în spaţiu a liniilor de contur 203
21.1.5. Reprezentarea grafică a câmpurilor de vectori orientaţi 204
21.2. Reprezentări spaţiale cu linii 205
21.2.1. Reprezentarea liniilor în spaţiu 205
21.2.2. Reprezentarea grafică spaţială a poliedrelor 206
21.2.3. Reprezentări grafice spaţiale dinamice 206
21.3. Reprezentarea 3D a suprafeţelor şi liniilor de contur 207
21.3.1. Reprezentarea suprafeţelor cu „mesh" 208
21.3.2. Reprezentarea grafică a suprafeţelor netede 209
21.4. Reprezentarea obiectelor spaţiale 210
21.4.1. Reprezentarea grafică a obiectului cilindru 210
21.4.2. Reprezentarea grafică a obiectului sferă 211
21.5. Poziţionarea observatorului faţă de obiect 212
21.5.1. Definirea poziţiei observatorului faţă de obiect 212
21.5.2. Vizualizarea şi ascunderea suprafeţelor suprapuse 213

22. PERSONALIZAREA GRAFICELOR 214
22.1. Precizarea titlului graficului 214
22.2. Precizarea etichetelor axelor 214
22.3. Plasarea unui text pe grafic 214
22.4. Trasarea unei reţele („grid") pe grafic 215

23. CREAREA ŞI CONTROLUL AXELOR 216
23.1. Divizarea ferestrei grafice 216
23.2. Schimbarea limitelor axelor 216
23.3. Suprapunerea succesivă a graficelor 218
23.4. Ştergerea axelor curente 218
23.5. Determinarea identificatorului axelor curente 218
23.6. Crearea unui obiect grafic axe 218
23.6.1. Proprietăţile generale ale obiectului grafic axe 220
23.6.2. Proprietăţile de control ale axelor (X, Y şi Z) 223

24. CREAREA ŞI CONTROLUL FIGURILOR 225
24.1. Crearea unui obiect figură 225
24.2. Determinarea identificatorului unei figuri 225
24.3. Proprietăţile obiectului figură 226
24.4. Ştergerea unei figuri 230
24.5. Închiderea unei figuri 230

25. CREAREA UNOR OBIECTE GRAFICE 231
25.1. Crearea unui obiect grafic text 231
25.1.1. Proprietăţile obiectului grafic text 231
25.2. Crearea unui obiect grafic linie 233
25.2.1. Proprietăţile obiectului grafic linie 234
25.3. Crearea unui obiect grafic suprafaţă 235
25.3.1. Proprietăţile obiectului grafic suprafaţă 236
25.4. Crearea unui obiect grafic imagine 237
25.4.1. Proprietăţile obiectului grafic imagine 238

26. CONTROLUL CARACTERISTICILOR GRAFICE 239
26.1. Stabilirea proprietăţilor obiectelor 239
26.2. Citirea proprietăţilor obiectelor 240
26.3. Iniţializarea proprietăţilor obiectelor 242
26.4. Actualizarea şi ştergerea obiectelor 242

27. CONTROLUL CULORILOR REPREZENTĂRILOR GRAFICE 243
27.1. Controlul culorilor reprezentărilor grafice 243
27.1.1. Specificarea culorii în MATLAB 243
27.1.2. Stabilirea matricei de culoare 244
27.1.3. Scalarea matricei de culoare 244
27.1.4. Stabilirea modului de colorare 246
27.1.5. Matricea de pseudoculoare 246
27.1.6. Conversia culorilor RGB-HSV şi HSV-RGB 247
27.1.7. Controlul contrastului şi luminozităţii 248
27.1.8. Rotirea culorii reprezentării grafice 248
27.1.9. Reprezentarea grafică a ponderii culorilor fundamentale 249
27.2. Controlul iluminării reprezentărilor grafice 249
27.2.1. Reprezentarea grafică a normalei la suprafaţă 250
27.2.2. Controlul nuanţelor de culoare prin surse de lumină 250
27.2.3. Reprezentarea suprafeţelor cu reflexii 251
27.3. Modele de culoare folosite în reprezentările grafice 251
27.3.1. Modelul „bone" 252
27.3.2. Modelul „cool" 252
27.3.3. Modelul „copper" 252
27.3.4. Modelul „flag" 252
27.3.5. Modelul „gray" 253
27.3.6. Modelul „hot" 253
27.3.7. Modelul „hsv" 253
27.3.8. Modelul „jet" 254
27.3.9. Modelul „pink" 254
27.3.10. Modelul „prism" 254

28. MIŞCAREA ŞI ANIMAŢIA IMAGINILOR 255

29. ÎNREGISTRAREA REPREZENTĂRILOR GRAFICE 257

30. CREAREA INTERFEŢELOR GRAFICE INTERACTIVE 259
30.1. Crearea elementelor de control pentru interfeţele grafice interactive 259
30.1.1. Proprietăţile elementelor de control ale interfeţelor grafice interactive 260
30.2. Crearea meniurilor pentru interfeţele grafice 264
30.2.1. Proprietăţile obiectului meniu 265
30.3. Citirea interactivă a datelor de pe grafice 266
30.4. Selectarea interactivă a fişierelor citite 267
30.5. Stabilirea interactivă a fişierelor salvate 268
30.6. Selectarea interactivă a fonturilor 269
30.7. Selectarea interactivă a culorilor 270

Partea a IV-a. APLICAŢII

31.1. Semnale cu ecouri 271
31.2. Semnale aleatoare cu distribuţie uniformă 272
31.3. Semnale aleatoare cu distribuţie normală 274
31.4. Raportul semnal - zgomot 275
31.5. Curenţii într-un circuit electric 276
31.6. Circuit electric în regim tranzitoriu 277
31.7. Simularea numerică a fiabilităţii 279
31.8. Evaluarea prin interpolare a temperaturii 281
31.9. Temperatura de echilibru a unei plăci 282
31.10. Pătrunderea căldurii în semispaţiul omogen 285
31.11. Timpul de topire al unui fir fuzibil 287
31.12. Câmpul magnetic armonic într-un conductor cilindric 289
31.13. Interfaţă grafică interactivă 291

ANEXĂ. Indexul funcţiilor MATLAB 299

BIBLIOGRAFIE 302

Partea I
INTRODUCERE ÎN MATLAB

1. CE ESTE MATLAB ?

MATLAB este un pachet de programe de înaltă performanţă, dedicat calculului numeric şi reprezentărilor grafice în domeniul ştiinţei şi ingineriei.

El integrează analiza numerică, calculul matriceal, procesarea semnalului şi reprezentările grafice, într-un mediu uşor de învăţat şi folosit, în care enunţurile problemelor şi rezolvările acestora sunt exprimate în modul cel mai natural posibil, aşa cum sunt scrise matematic, fără a fi necesară programarea tradiţională. Elementul de bază cu care operează MATLAB-ul este matricea. Cu acesta se pot rezolva probleme fără a fi necesară scrierea unui program într-un limbaj de programare.

Dezvoltat de-a lungul mai multor ani, acum MATLAB-ul este un standard în mediile universitare, precum şi în domeniile cercetării şi rezolvării practice a problemelor legate de procesarea semnalelor, identificarea sistemelor, controlul statistic, prelucrarea datelor experimentale etc.

Dintre resursele de calcul şi reprezentare grafică ale MATLAB-ului se menţionează următoarele:

1. Operaţii matematice fundamentale

◊ calcule cu matrice şi tablouri;
◊ operatori relaţionali şi logici;
◊ funcţii trigonometrice şi hiperbolice, exponenţiala, logaritmul etc;
◊ funcţiile speciale: Gamma, Bessel, Euler, eliptică etc.;
◊ calcule cu polinoame;
◊ aritmetică în virgulă mobilă IEEE (15 cifre zecimale în rezultat).

2. Algebră liniară şi funcţii matriceale

◊ analiza matriceală;
◊ ecuaţii liniare;
◊ descompunerea în valori proprii şi valori singulare;
◊ factorizarea QR;
◊ generarea matricelor;
◊ algoritmi pentru matrice rare.

3. Analiza datelor şi transformatele Fourier

◊ corelaţia, covarianţa, diferenţe finite;
◊ transformata Fourier rapidă (FFT) ;
◊ media statistică, mediana şi deviaţia standard;
◊ interpolarea datelor.

4. Analiză numerică neliniară
- ◊ ecuaţii diferenţiale;
- ◊ integrarea numerică;
- ◊ aproximarea trecerilor prin zero;
- ◊ minimizarea.

5. Programare
- ◊ structuri de control;
- ◊ citirea/scrierea din/în fişiere format ASCII sau binar;
- ◊ depanarea fişierelor program;
- ◊ crearea de interfeţe grafice interactive (GUI).

6. Reprezentarea graficelor 2D şi 3D
- ◊ folosirea pixelilor, liniilor, poligoanelor pline, liniilor de contur;
- ◊ utilizarea diagramelor polare, de bare sau a histogramelor;
- ◊ reprezentarea în coordonate liniare, semilogaritmice şi logaritmice;
- ◊ reprezentarea suprafeţelor.

7. Vizualizarea graficelor
- ◊ controlul culorilor şi a poziţiei sursei de lumină;
- ◊ vizualizări volumetrice;
- ◊ afişarea imaginilor;
- ◊ animaţie.

8. Proprietăţi şi resurse grafice
- ◊ crearea şi manipularea figurilor;
- ◊ desenarea, scalarea, etichetarea şi colorarea axelor;
- ◊ desenarea şi manipularea liniilor şi textului.

9. Utilizarea interfeţei grafice (Graphical User Interface - GUI)
- ◊ crearea de meniuri pentru bara superioară (pull-down) şi meniuri icoană (pop-up);
- ◊ crearea de butoane selectabile prin apăsare (push buttons), butoane radio (radio buttons), cutii de dialog (check boxes) şi cursoare liniare (sliders);
- ◊ cutii de dialog cu texte editabile sau fixe;
- ◊ utilizarea mouse-ului la detectarea unui eveniment sau a unei condiţii impuse, sau pentru lansarea unei comenzi.

10. Suportul pentru imprimare
- ◊ imprimante alb-negru şi color, nivelul 1 şi nivelul 2 Post Script;
- ◊ familia de imprimante HP LaserJet;
- ◊ imprimante HP DeskJet, HP PaintJet;
- ◊ imprimante Epson cu 9 sau 24 pini.

11. Schimbul de documente
- ◊ importul şi exportul de fişiere în/din format ASCII, binar sau hexa;
- ◊ transmiterea de grafice în formatul standard META;
- ◊ compatibilitate cu formatul Clipboard.

Cea mai importantă caracteristică a MATLAB-ului este uşurinţa cu care poate fi extins. Prin aceasta, orice utilizator poate adăuga propriile programe scrise în MATLAB la fişierele originale, dezvoltând aplicaţii specifice domeniului în care lucrează.

De asemenea, MATLAB-ul include aplicaţii specifice, numite TOOLBOX-uri. Acestea sunt colecţii extinse de funcţii MATLAB (fişiere M) care dezvoltă mediul de programare de la o versiune la alta, pentru a rezolva probleme din domenii variate. Structural, MATLAB-ul este realizat sub forma unui nucleu de bază, cu interpretor propriu, în jurul căruia sunt construite toolbox-urile. Până la această versiune, firma The MATH WORKS Inc. a pus în circulaţie următoarele toolbox-uri:

1. **SIGNAL PROCESSING** - recomandat pentru procesarea semnalelor şi analiza seriilor temporale; cuprinde următoarele grupe de funcţii:
- implementarea şi proiectarea filtrelor analogice şi digitale;
- analiza şi estimarea spectrului;
- simularea răspunsului filtrelor;
- transformatele FFT, DCT etc. ;
- modelare parametrică;
- modulare şi demodulare.

2. **IMAGE PROCESSING** - este o colecţie de funcţii avansate pentru procesarea imaginilor şi a semnalelor bidimensionale; cuprinde următoarele grupe de funcţii:
- proiectarea filtrelor 2-D şi filtrarea semnalelor bidimensionale;
- reconstituirea şi extragerea imaginilor;
- operaţii de colorare, geometrie şi morfologie;
- transformări bidimensionale;
- analiza şi statistica imaginilor.

3. **SYMBOLIC MATH** - este un nucleu integrat, bazat pe pachetul software MAPLE V; cuprinde următoarele grupe de funcţii:
- calculul simbolic pentru rezolvarea problemelor de algebră liniară şi a ecuaţiilor;
- simplificarea simbolică a expresiilor;
- evaluarea simbolică a funcţiilor matematice speciale;
- accesul la nucleul MAPLE, includerea librăriilor şi a pachetului de algebră liniară.

4. **NEURAL NETWORK** - pentru proiectarea şi simularea reţelelor neurale; cuprinde următoarele grupe de funcţii:
- funcţia de transfer sigmoidă, liniară, limită şi concurenţială;
- reţele asociative, cu punct de oprire, cu transformare de caracteristici, cu auto-organizare, Hopfied, Kohonen, Widrow-Hoff;
- arhitecturi cu estimări recurente, succesive;
- funcţii pentru analiza grafică a rezultatelor;
- straturi, elemente şi conexiuni nelimitate.

5. **STATISTICS** - este o colecţie de funcţii folosite pentru analiza, modelarea şi simularea datelor care conţine:
- analiza grafică interactivă (GUI);

- distribuţiile beta, binomială, hi-pătrat, Poisson etc. ;
- generarea numerelor aleatoare;
- calcule de regresie, polinomiale şi varianţă;
- descrieri statistice;
- teste de ipoteză;
- analiza interactivă a liniilor de contur.

6. **SPLINE** - aproximează şi modelează datele prin intermediul funcţiilor:
- interpolări polinomiale şi funcţii B-spline;
- construirea funcţiilor spline şi operarea cu acestea;
- aproximarea curbelor şi netezirea acestora;
- calculul derivatei, integralei şi evaluarea funcţiilor.

7. **CONTROL SYSTEM DESIGN** - este utilizat pentru analiza şi proiectarea sistemelor automate de control şi conţine:
- tehnici moderne şi clasice;
- sisteme în timp continuu şi în timp discret;
- spaţiul stărilor şi funcţiile de transfer ale modelelor;
- interconectarea sistemelor;
- transformări între modele;
- răspunsul în frecvenţă: Bode, Nyquist, Nichols, SVD;
- locul rădăcinilor, plasarea polilor, LQG.

8. **ROBUST CONTROL** - este un instrument pentru sinteza sistemelor care trebuie să funcţioneze în prezenţa incertitudinii şi conţine:
- sinteza controlului optimal LQG/LTR;
- sinteza controlului optimal H_2 şi H_∞;
- reducerea modelului la valori singulare;
- factorizarea spectrală şi construirea modelelor.

9. **SYSTEM IDENTIFICATION** - este utilizat pentru procesarea semnalului în vederea modelării parametrice, a identificării sistemelor şi a analizei seriilor temporale. Abordează următoarele probleme:
- modelarea MA, AR, ARMA şi Box-Jenkins;
- spaţiul stărilor şi funcţiile de transfer ale modelelor;
- validarea modelului şi a celei mai bune aproximări;
- selectarea automată sau manuală a ordinului modelului;
- modelarea bazată pe procesarea semnalului;
- analiza spectrală.

10. **μ-ANALYSIS AND SYNTHESIS.** Este recomandat pentru μ-sinteza şi proiectarea sistemelor în prezenţa incertitudinii prin:
- μ-analiză şi sinteză;
- sinteza H_2 şi H_∞.

11. **OPTIMISATION** - este o colecţie de funcţii utilizată pentru optimizarea liniară şi neliniară; grupează probleme de:
- programare liniară şi programare pătratică;
- determinarea minimului şi maximului;
- funcţii neliniare rezolvate în sensul celor mai mici pătrate;
- optimizarea constrângerilor;
- rezolvarea ecuaţiilor neliniare;
- rezolvarea problemelor de minimax şi semi-infinite;
- optimizarea multiobiectiv.

Versiunea **MATLAB** descrisă în această lucrare este **MATLAB4 sub WINDOWS.** Deşi majoritatea funcţiilor menţionate în continuare funcţionează şi cu versiuni mai vechi, funcţiile grafice pot pune anumite probleme.

2. MATLAB SUB WINDOWS

Începând cu versiunea 4.0 din 1992, **MATLAB** rulează sub **WINDOWS** într-o prezentare nouă, orientată spre ferestre şi meniuri. Această nouă versiune, care face posibilă vizualizarea mai multor ferestre grafice în acelaşi timp, adaptată stilului de operare sub sistemul WINDOWS, menţine compatibilitatea cu programele scrise în versiunile anterioare. Faţă de versiunile **DOS**, versiunile sub WINDOWS au un număr semnificativ de funcţii noi, atât pentru calcul, cât mai ales pentru reprezentările grafice. Acestea sunt atât de mult perfecţionate, încât au dezvoltat mediul de programare MATLAB până la posibilitatea de a-l utiliza la procesarea semnalelor bidimensionale şi a imaginilor. Partea de documentare, inclusă în structura **Help** a produsului este aliniată sistemului WINDOWS. Există posibilităţi de căutare a explicaţiilor atât prin index, cât şi prin cuprins (grupe de funcţii - **Table of Contents**).

2.1. Instalarea

Calculatorul pe care pot rula versiunile de **MATLAB sub WINDOWS** trebuie să aibă cel puţin procesor 386 şi coprocesor 387, un spaţiu liber pe hard disc de 15 Mo şi minim 4 Mo RAM de memorie. Dacă se doreşte utilizarea animaţiei, atunci este necesar ca monitorul să fie Super VGA cu cel puţin 2 Mo de memorie video. **MATLAB sub WINDOWS** se livrează pe 4 dischete de 1,44 Mo şi ocupă circa 10 Mo (fără toolbox-uri). Produsul necesită **Windows 3.0** sau versiuni ulterioare. Instalarea se face prin lansarea programului INSTALL.EXE pe de prima dischetă, de sub mediul Windows. După lansarea instalării se urmăresc mesajele care vor conduce la reuşita operaţiei, după introducerea datelor de identificare: numărul licenţei, firma care-l utilizează şi numele utilizatorului. Dacă se dispune de dischete cu toolbox-uri (module speciale care sunt extensii ale pachetului de programe MATLAB, dedicate unei grupe de aplicaţii), acestea se introduc în continuare, construindu-se automat directorii corespunzători. La sfârşitul procesului de instalare se creează automat o pictogramă cu numele **„MATLAB for WINDOWS"** (dacă nu se specifică altfel).

2. 2. Lansarea în execuţie

Programul se lansează în execuţie din mediul Windows, prin selecţia pictogramei MATLAB. Din afara mediului Windows (de exemplu, DOS), MATLAB-ul se poate lansa direct cu o comandă de forma:

win d:\MATLAB\BIN\matlab

dacă fişierul **„matlab.exe"** se găseşte în calea **„MATLAB\BIN"** din unitatea **d**. La lansare, va apărea pe ecran sigla **„MATLAB for WINDOWS"** cu înscrierea versiunii, a licenţei şi a utilizatorului.

2.3. Ferestrele de lucru

MATLAB-ul lucrează cu două tipuri de ferestre: o fereastră de comenzi şi o fereastră pentru reprezentări grafice.

2.3.1. Fereastra de comenzi

Fereastra de comenzi este prezentată în figura 2.1. Meniul din bara superioară este accesibil prin tastarea simultană a tastei **[Alt]** şi a literei subliniate a comenzii dorite sau prin selecţia directă cu mouse-ul a comenzii dorite.

Fiecare comandă din meniul principal furnizează un meniu specific, în cadrul acestuia selecţia putându-se face fie cu mouse-ul, fie cu ajutorul săgeţilor prin deplasarea zonei active, fie prin tastarea literei marcate special în fiecare subcomandă a respectivului meniu.

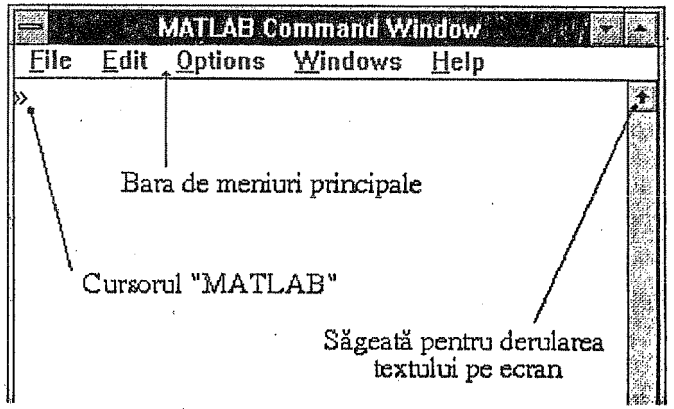


Fig. 2.1. Fereastra de comenzi a MATLAB-ului sub WINDOWS

2. 3.1.1. Gestionarea fişierelor

Selectând comanda **File** din meniul principal, prin **[Alt]+F**, sau prin poziţionarea mouse-ului în dreptul cuvântului **File**, se obţine un submeniu ca cel prezentat în figura 2.2. Literele subliniate corespund comenzii căreia îi aparţin, iar acţionarea tastei respective activează comanda corespondentă.

New provoacă deschiderea unui alt sub-submeniu, care are opţiunile **„M-file"** şi **„Figure"**. Alegerea opţiunii **„M-file"** va deschide o fereastră de editare a unui fişier M, în timp ce opţiunea **„Figure"** deschide o fereastră grafică nouă.

Open M file... deschide o fereastră de dialog pentru a selecta un fişier care va deveni fişier de lucru.

Look for Selected şi **Open Selected** analizează fişierele pentru selectare şi deschide pe cel selectat.

Save Workspace As... deschide o fereastră de dialog pentru a salva datele din spaţiul de lucru într-un fişier, al cărui nume trebuie precizat.

Print ... şi **Printer Setup....** Comenzile de tipărire a documentului, Print şi Printer Setup, apar în acelaşi meniu de gestionare a fişierelor. Comanda „Print..." permite tipărirea documentului, iar „Printer Setup..." configurează pagina, marginile, tipul de imprimantă etc.

Exit MATLAB. În ultima poziţie din meniul File apare această comandă ce are ca efect părăsirea aplicaţiei. Comanda Exit MATLAB poate fi selectată de asemenea tastând **[Alt]+x** din exteriorul meniului.

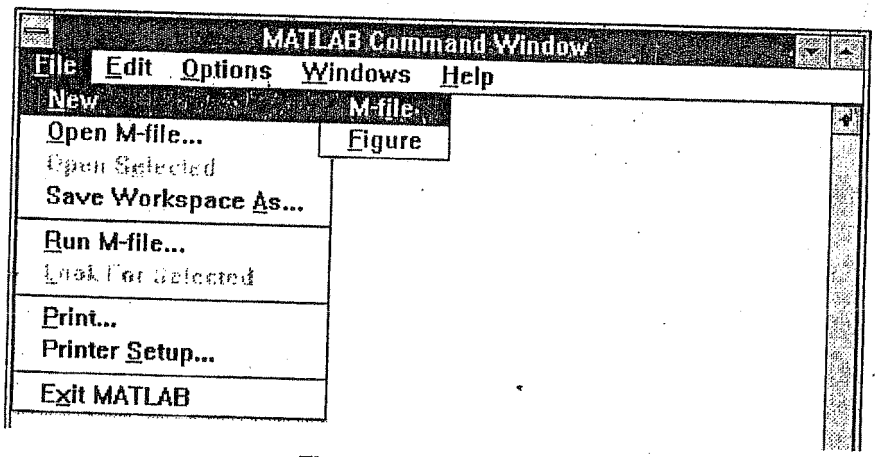


Fig. 2.2. Submeniul **File.**

2. 3.1.2. Editarea programelor

Selectând comanda **Edit** din meniul principal, apare submeniul prezentat în figura 2.3.

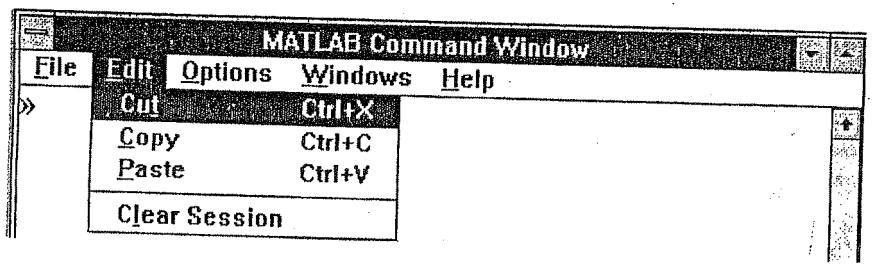


Fig. 2.3. Submeniul **Edit.**

Modul de lucru în acest subdomeniu este identic cu cel din Windows sau din cel al unui editor de text care lucrează sub Windows. După selectarea unui text, din mediul MATLAB sau din alt document, acesta se poate folosi cu opţiunile **Cut** (Taie), **Copy** (Copiază) sau **Paste** (Lipeşte). Ştergerea sesiunii de lucru se realizează cu comanda **Clear Session.**

2. 3.1.3. Alegerea opţiunilor

Selectând comanda **Options** din meniul principal, se obţine submeniul prezentat în figura 2.4.

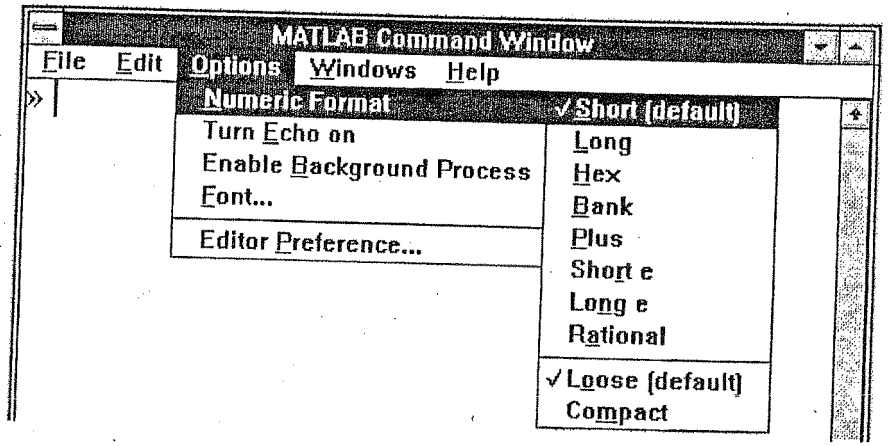


Fig. 2.4. Submeniul **Options.**

Numeric Format este un submeniu care permite selectarea unuia dintre formatele de afişare: **Short** (5 cifre) / **Long** (15 cifre) / **Hex** (hexazecimal)/ **Bank** / **Plus** (+, - şi blanc)/ **Short e** (5 cifre + exponenţiala)/ **Long e** (15 cifre+exponenţiala)/ **Rational** (fracţii), precum şi a distanţelor dintre rândurile succesive: **Loose** (distanţat)/ **Compact** (strâns).

Turn Echo on este un comutator (on/off) care permite afişarea liniilor programului MATLAB, în timpul rulării acestuia.

Enable Background process este un comutator (on/off) care permite rularea programelor MATLAB, în timp ce calculatorul poate efectua şi alte operaţii.

Font... deschide o fereastră de dialog pentru selectarea fonturilor (tip, mărime şi culoare) care se vor utiliza la afişarea pe ecran a sesiunii de lucru MATLAB.

Editor Preference... selectează editorul de texte cu care se vor scrie programele MATLAB. Dacă nu se specifică altfel, se alege implicit **NOTEPAD.EXE.**

2. 3.1.4. Selectarea ferestrelor

Comanda **Windows** din meniul principal afişează submeniul prezentat în figura 2.5.

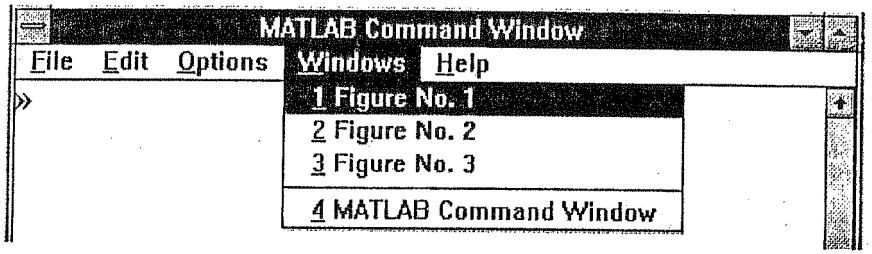


Fig. 2.5. Submeniul **Windows.**

Trecerea dintr-o fereastră grafică sau de comenzi într-o altă fereastră se realizează prin selectarea comenzii **Windows** din meniul principal, urmată de selectarea uneia dintre ferestrele grafice deschise (ex: **Figure No.1, Figure No.2,...**) sau a celei de comenzi (**MATLAB Command Window**).

2. 3.1.5. Documentaţia de ajutor - Help

Comanda **Help** din meniul principal afişează submeniul prezentat în figura 2.6.

Table of Contents. Afişează o listă senzitivă cu toţi subdirectorii din corpul principal al MATLAB-ului. La plasarea săgeţii mouse-ului pe unul din subdirectorii listaţi, aceasta se transformă într-o **„mână"** care permite, prin selectare, afişarea help-ului grupei de funcţii respective.

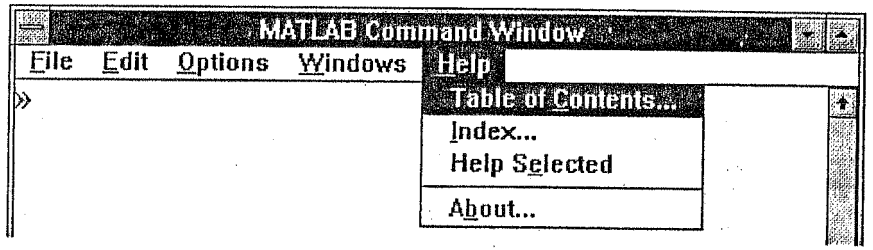


Fig. 2.6. Submeniul **Help.**

Index furnizează o listă cu subdirectorii principali ai MATLAB-ului. Selectarea subdirectorului dorit afişează conţinutul acestuia şi apoi informaţii despre un anumit fişier sau funcţie.

Help Selected afişează documentaţia Help pentru funcţia MATLAB ce a fost selectată cu mouse-ul.

2.3.2. Fereastra de reprezentări grafice în MATLAB

Fereastra grafică este o formă elevată de reprezentare a graficelor; pot exista mai multe ferestre grafice deschise în acelaşi timp, dar o singură fereastră de comenzi.

2.3.2.1. Gestionarea ferestrelor grafice

New Figure sau **New / Figure.** Selectarea comenzii **File** (din fereastra grafică), urmată de comenzile **New** şi **Figure** sau numai de **New Figure,** determină trecerea într-o fereastră grafică nouă (figura 2.7.). Dacă se selectează comanda **Windows** (din meniul principal), urmată de opţiunea **Figure** (ex. Figure No.1), se trece în fereastra grafică ce conţine o reprezentare anterioară (figura 2.5). În fereastra grafică pot fi reprezentate funcţii 2D sau 3 D, imagini color, animaţie etc.

Close închide fereastra grafică curentă.

2. 3.2.2. Editarea graficelor

Selectând comanda **Edit** din meniul principal al ferestrei grafice, apare submeniul prezentat în figura 2.8.

Copy to Metafile copiază figura curentă în format meta (vezi formatele fişierelor grafice în Windows)

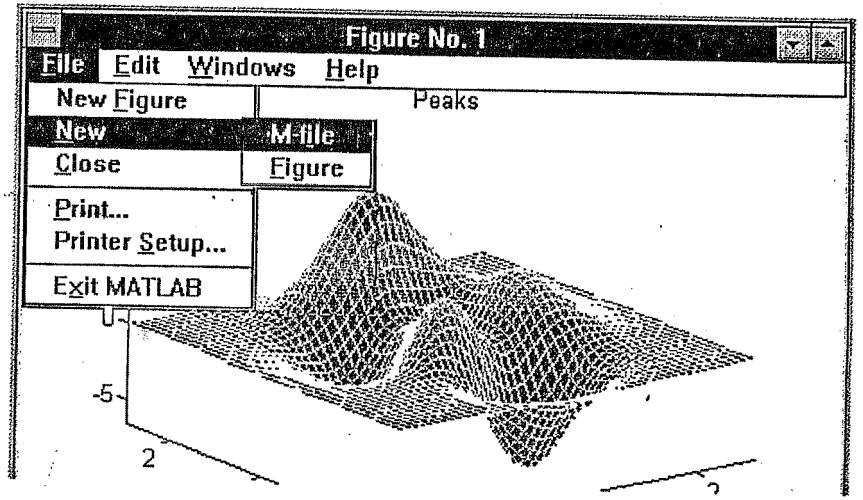


Fig. 2.7. Submeniul **File** al ferestrei grafice.

Copy to Bitmap copiază figura curentă în format bitmap. (vezi formatele fişierelor grafice în Windows)

Clear Figure şterge figura curentă, fără să o închidă.

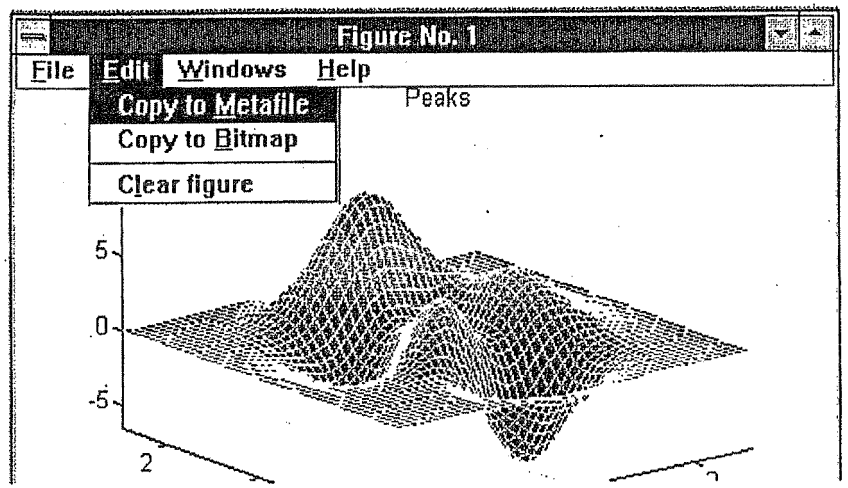


Fig. 2.8. Submeniul **Edit** al ferestrei grafice a MATLAB-ului.

3. FUNCŢII MATLAB DE INTERES GENERAL

3.1. Funcţii pentru lansarea şi ieşirea din MATLAB

Funcţiile pentru lansarea unei sesiuni de lucru în MATLAB, părăsirea acesteia, precum şi pentru asigurarea unor informaţii necesare unei funcţionări corecte sunt:

startup Fişier M, executat la activarea MATLAB-ului;

matlabrc Fişierul principal pentru lansarea MATLAB-ului (conţine toate căile de acces la fişierele de lucru, precum şi setările parametrilor de lucru pentru mediul de programare MATLAB);

exit, quit Comandă pentru ieşirea din MATLAB.

Lansarea în execuţie a MATLAB-ului apelează automat fişierul **matlabrc.m**. Pe un calculator pe care au acces mai mulţi utilizatori sau pe reţele de calculatoare acest fişier este rezervat managerului de sistem. Dacă fişierul **matlabrc.m** conţine instrucţiunile:

```
if exist('startup')==2
      startup
end
```

atunci este apelat şi fişierul **startup.m**. Acest procedeu permite crearea unor fişiere-M adiţionale, care pot fi apelate la lansare, fie pentru citirea unor parametri, scale de etalonare, coeficienţi de corecţie etc, fie pentru verificarea unor condiţii care trebuie îndeplinite pentru ca programul lansat să se execute corect. De asemenea, constantele fizice, factorii de conversie ai unităţilor de măsură sau orice altă constantă predefinită într-o aplicaţie se pun în aceste fişiere.

Părăsirea sesiunii de lucru se face fie prin selectarea succesivă a opţiunilor **File/ExitMATLAB**, fie prin scrierea explicită în fereastra de comenzi a comenzilor **„exit"** sau **„quit"** (fără apostrofuri). Acest procedeu de părăsire a unei sesiuni de lucru determină pierderea variabilelor din aplicaţia curentă.

3.2. Funcţii de control general

Pentru controlul general al mediului de programare MATLAB se utilizează funcţiile:

help Furnizează informaţii „on-line" despre MATLAB şi funcţiile acestuia;

what Listează fişierele *.M, *.MAT şi *.MEX din directorul curent;

type Listează fişierul M, menţionat;

lookfor Returnează numele fişierelor care au în prima linie a HELP-ului (linia H1) cuvintele precizate ca argument;

which Returnează calea în care este localizat un fişier sau o funcţie MATLAB;

path Returnează căile cu care lucrează MATLAB-ul (în care sunt căutate fişierele apelate);

who Listează variabilele curente din memorie.

whos Listează variabilele curente, dimensiunile lor, precum şi tipul acestora (reale sau complexe);

exist Verifică dacă o variabilă există în mediul MATLAB.

3.2.1. Help-ul „on-line"

Funcţiile din această secţiune permit obţinerea informaţiilor de interes general referitoare la mediul de lucru MATLAB. Cu comanda **help** se pot obţine informaţii ajutătoare despre o anumită funcţie, se pot lista fişierele dintr-un director sau se poate lista conţinutul unui anumit fişier etc.

Funcţia *help* furnizează informaţii „on-line" pentru orice funcţie MATLAB sau realizată de un utilizator al pachetului de programe. Lista funcţiilor se obţine tastând:

```
help
```

iar informaţiile specifice se obţin tastând:

```
help subiect
```

„Subiect" poate fi un nume de funcţie sau un nume de director. Dacă acesta este nume de funcţie, *help*-ul afişează informaţii despre funcţie. Dacă acesta este nume de director, *help*-ul afişează fişierele conţinute în directorul specificat. La apelare nu este necesar să se precizeze întreaga cale a directorului: ultima componentă sau ultimele câteva componente sunt suficiente. Prin acest procedeu se pot scrie help-uri pentru fişierele - M sau toolbox-urile proprii utilizatorului. Sistemul de help al MATLAB-ului este foarte uşor de extins. Prin acest mijloc se pot scrie help-uri pentru descrierea unui fişier - M sau toolbox, la fel de uşor de apelat ca cele furnizate de producătorul pachetului de programe.

La apelarea funcţiei *help* fără parametri se afişează prima linie (**linia H1**) a fişierelor conţinute în fiecare director din calea de căutare. Fişierele - M conţinute în fiecare director sunt citite din fişierul **contents.m** al acestuia.

Instrucţiunea *help* subiect, în care „subiect" este numele unui director, afişează liniile de comentariu ale fişierului **contents.m** localizat în acel director. Dacă fişierul contents.m nu are comentariu, funcţia *help* afişează toate fişierele conţinute în director. Dacă „subiect" este numele unei funcţii, instrucţiunea *help* afişează liniile primului comentariu al fişierului-M, numit **subiect.m.** Crearea unui help „on-line" pentru un fişier-M sau pentru un director se face ca în exemplul detaliat în §4.8. Spre exemplu, instrucţiunea:

```
help help
```

returnează informaţii despre modul de utilizare al help-ului, iar:

```
help what
```

prezintă help-ul funcţiei MATLAB *what*.

3.2.2. Informaţii despre fişierele unui director

Funcţia *what* returnează informaţii referitoare la fişierele *.m, *.mat şi *.mex din directorul curent. Dacă se apelează cu un nume de director (sau numai ultimul subdirector), se afişează aceeaşi grupă de fişiere conţinute în directorul (subdirectorul) din calea specificată. Spre exemplu:

```
what general
```

returnează:

```
M-files in directory c:\matlab\toolbox\matlab\general
cd     diary   getenv  length  memory  readme  unix    whos
cedit  dir     help    load    more    save    ver
.................................
```


adică numele fişierelor din subdirectorul **general.**

Funcţia *which* returnează calea (director\subdirector) în care se găseşte un anumit fişier, menţionat ca argument. Astfel, instrucţiunea:

```
which demo
```

returnează:

```
c:\matlab\toolbox\matlab\demos\demo.m
```

adică localizarea fişierului **demo.m.**

Funcţia *lookfor* listează toate numele de fişiere din toţi directorii care au în prima linie a help-ului cuvintele menţionate ca argument, precum şi acea linie. Se apelează cu sintaxa:

```
lookfor  cuvinte_căutate_în _linia_de_help
```

Spre exemplu, instrucţiunea:

```
lookfor ifft
```

caută în help-ul tuturor fişierelor din toate directoarele din căile de lucru MATLAB cuvântul *ifft*, şi returnează:

```
IFFT     Inverse discrete Fourier transform.
IFFT2    Two-dimensional inverse discrete Fourier transform.
```

care sunt numele fişierelor şi prima linie a help-ului în care se găseşte cuvântul *ifft*.

3.2.3. Afişarea conţinutului unui fişier

Afişarea unui fişier pe ecran, fără a-l putea modifica, se face cu funcţia *type*; se apelează cu sintaxa:

```
type nume_fisier
```

Comanda *type* consideră că fişierul apelat are extensia *.m, dacă aceasta nu este precizată. Dacă se doreşte afişarea unui fişier cu altă extensie, atunci aceasta trebuie precizată la apelare.

3.2.4. Căile de căutare

Funcţia *path* controlează căile de căutare ale MATLAB-ului. Spre exemplu la tastarea cuvântului „semnal", MATLAB-ul procedează în următoarea ordine:

1. caută în memorie variabila cu numele „semnal";
2. caută în memorie funcţia cu numele „semnal";
3. caută în directorul curent de pe disc fişierul cu numele „semnal.m";
4. caută în directorii specificaţi în *path* fişierul cu numele „semnal.m".

Funcţia *path* constituie mediul variabil al sistemului de operare, în care se pot predefini, pot fi schimbaţi sau pot fi adăugaţi directorii proprii în calea de căutat.

Instrucţiunea:

```
path
```

afişează setarea căilor de căutare curente ale MATLAB-ului. Lista căilor de căutare este setată în fişierele **startup.m** sau **matlabrc. m.** Apelată cu sintaxa:

```
p=path
```

funcţia returnează în variabila p calea curentă de căutare. Dacă se doreşte schimbarea căii de căutare, la una precizată în variabila p, atunci se apelează cu sintaxa:

```
path(p)
```

Adăugarea unor căi noi p2, la cele existente p1 (concatenarea căilor de căutare p1 cu p2) se face cu instrucţiunea

```
path(p1, p2)
```

De asemenea, *path*(*path*,p) adaugă calea p la cele existente, iar *path*(p,*path*) renunţă la calea p. Astfel, instrucţiunea :

```
path(path,'c:\MATLAB\TEST')
```

adaugă o nouă cale de căutare (c:\MATLAB\TEST) la cele deja existente.

3.2.5. Informaţii despre variabilele dintr-un fişier

Instrucţiunile introduse într-o sesiune MATLAB creează variabile care sunt memorate în spaţiul de lucru şi pot fi identificate cu funcţiile *who* şi *whos*.

Funcţia *who* listează numele variabilelor din spaţiul de lucru; se apelează cu sintaxa:

```
who
```

şi returnează:

Your variables are:
D XF ans p x
.....

Funcţia *whos* furnizează în plus informaţii suplimentare referitoare la dimensiunea fiecărei variabile curente; se apelează cu sintaxa:

```
whos
```

şi returnează:

Name	Size	Elements	Bytes	Density	Complex
D	1 by 8	8	64	Full	No

.....

Funcţia *exist* se apelează cu sintaxa:

```
exist('nume_variabilă')
```

şi returnează:

0 - dacă „nume_variabilă" nu există;
1 - dacă „nume_variabilă" este o variabilă în spaţiul de lucru;
2 - dacă „nume_variabilă" este un fişier-M într-una din căile MATLAB;
3 - dacă „nume_variabilă" este un fişier-MEX într-una din căile MATLAB;
4 - dacă „nume_variabilă" este o funcţie SIMULINK;
5 - dacă „nume_variabilă" este o construcţie internă MATLAB.
Numele variabilei trebuie să fie scris între apostrofuri (').

3.3. Funcţii pentru controlul directoarelor, fişierelor şi al sistemului de operare

cd	Returnează sau schimbă directorul curent;
dir	Listează conţinutul directorului curent;
delete	Şterge fişierul precizat, de pe disc;
!	Execută comenzi din sistemul de operare DOS, fără părăsirea MATLAB-ului;
diary	Salvează sesiunea de lucru (fără partea grafică) într-un fişier text (ASCII).

3.3.1. Controlul directoarelor şi fişierelor

Comenzile *dir*, *type*, *delete* şi *cd* implementează un set de comenzi generale ale sistemului de operare, care permit manipularea fişierelor de pe disc.

Funcţia *dir*, care se apelează cu sintaxa:

```
dir
```

afişează numele tuturor fişierelor din directorul curent sau din orice alt director precizat ca argument.

Funcţia *delete* permite ştergerea unui fişier sau a unui grafic; se apelează cu sintaxa:

delete nume_fişier

Funcţia *cd* returnează numele directorului curent, dacă se apelează cu sintaxa:

```
cd
```

sau schimbă directorul, dacă se apelează cu sintaxa:

cd cale/nume_director

Funcţia *diary* creează pe disc un fişier al sesiunii de lucru în MATLAB (fără grafice). Practic înregistrează toate comenzile şi răspunsurile returnate într-un fişier ASCII. Apelată cu sintaxa:

diary nume_fişier

funcţia realizează o copie a sesiunii de lucru, atât a comenzilor de intrare cât şi a răspunsurilor returnate; nume_fişier este numele fişierului în care se face înregistrarea. Dacă fişierul există deja, sesiunea de lucru se anexează la sfârşitul acestuia. Suspendarea înregistrării se realizează cu comanda:

```
diary off
```

iar reluarea înregistrării se face cu comanda :

```
diary on
```

Dacă nu a fost specificat nume_fişier, funcţia *diary* înregistrează sesiunea de lucru într-un fişier cu nume implicit **diary**.

3.3.2. Rularea programelor externe

Semnul de exclamare „!" plasat la începutul unei linii de program MATLAB indică mediului de programare că următoarele linii de intrare sunt introduse din sistemul de operare. Acest semn permite lansarea în execuţie a unor programe externe, fără a părăsi mediul MATLAB, precum şi executarea oricăror comenzi DOS (formatarea sau copierea unei dischete etc). De exemplu:

```
!tp
```

lansează mediul Turbo Pascal. La ieşirea din mediul sau programul apelat, controlul este returnat MATLAB-ului. Instrucţiunea:

```
!format A:
```

permite formatarea unei dischete care se află în unitatea A.

3.4. Funcţii de control al mărimilor de timp

clock	Funcţie pentru citirea ceasului calculatorului;
cputime	Funcţie pentru determinarea timpului de calcul al „CPU";
date	Funcţie pentru citirea datei din SETUP-ul calculatorului;
etime	Cronometrează intervalul de timp între două evenimente;
tic, toc	Funcţii pentru pornirea şi oprirea unui cronometru.

Funcţia *clock* returnează un vector linie de şase elemente conţinând timpul curent şi data în format zecimal: [an lună zi oră minut secundă].
Spre exemplu, cu instrucţiunea:

```
DATA=fix(clock)
```

se obţine rezultatul:

DATA= [1995 05 28 13 5 52]

care are semnificaţia: anul 1995, luna mai (5), ziua 28, ora 13, minutul 5, secunda 52.

Funcţia *date* returnează un şir de caractere conţinând data în formatul: zi-lună-an. De exemplu:

```
D=date
```

returnează: D= [28-May-1995]

Funcţia *etime* calculează un interval de timp; se apelează cu sintaxa:

etime(t1,t0)

unde momentele t1 şi t0 sunt momentele de timp final şi iniţial.

Exemplul 3.4.1. Să se scrie un program care să determine timpul necesar calculului transformatei Fourier rapide (FFT) în 2048 puncte.

Cu secvenţa de instrucţiuni MATLAB:

```
x = randn(2048,1);
t0 = clock;
      XF=fft(x); % secventa cronometrata
durata = etime(clock,t0)
```

se obţine răspunsul (în secunde): durata = 0.0600

3.5. Funcţii asociate ferestrei de comenzi

clc	Şterge fereastra de comenzi;
home	Mută cursorul în poziţia iniţială (prima linie - prima coloană);
format	Setează formatul de afişare a datelor;
echo	Permite afişarea liniilor de program în timpul executării acestora;
more	Funcţie pentru controlul numărului de linii afişate pe monitor.

3.5.1. Controlul ferestrei de comenzi

Afişarea succesivă a n linii dintr-un program sau matrice de date face apel la funcţia *more*; se apelează cu sintaxa:

more(n)

Cu tasta „ENTER" se trece la următoarea secvenţă de n linii, iar cu tasta Q se părăseşte modul de afişare cu număr de linii impus.

Funcţia de ştergere a ecranului, *clc*, se apelează cu sintaxa:

clc

iar funcţia *home* poziţionează cursorul în locaţia corespunzătoare primei linii şi primei coloane, apelându-se cu sintaxa:

home

3.5.2. Formate de afişare a numerelor

MATLAB-ul pune la dispoziţia utilizatorului două moduri de control al afişării datelor pe ecran: numărul de cifre şi spaţierea dintre ele.

Funcţia *format* stabileşte formatul extern de afişare al numerelor pe ecran. Această funcţie afectează doar modul de afişare a numerelor şi nu precizia calculului sau forma de salvare a datelor (MATLAB-ul execută calculele în dublă precizie).

La lansare, MATLAB afişează numerele în formatul scurt (cu 5 cifre). Trecerea de la un format la altul se poate face fie selectând succesiv :

Option/NumericFormat/opţiune

fie prin comanda:

format opţiune

unde „opţiune" poate fi una dintre variantele: [**Short] / Long / Short e / Long e / Hex / Bank / Plus / Rational**]. Formatele de afişare sunt:

OPŢIUNE	REZULTAT	EXEMPLE
short	5 cifre	1.3333
long	15 cifre	1.33333333333333
short e	5 cifre +exp.	1.3333E+000
long e	15 cifre + exp.	1.33333333333333E+000
hex	hexazecimal	3FF55555555555555
plus	plus blanc minus	+ -
bank	două zecimale	124.23
rat	exprimare raţională	102/53

Dacă toate elementele unei matrice sunt numere întregi, matricea este afişată într-un format fără zecimale. Atunci când cel puţin unul dintre elementele matricei nu este număr întreg, formatul de afişare respectă regulile prezentate mai sus, în funcţie de opţiunea selectată.

Pentru formate short şi long, dacă cel puţin un element al matricei este mai mare decât 1000 sau mai mic decât 0.001, afişarea acesteia se face cu un factor de scală comun pentru toate elementele.

Comanda *„format compact"* suprimă liniile libere şi afişează rezultatele într-un format mai compact, iar *„format loose"* afişează rezultatele într-un mod mai degajat (cu linii libere).

3.5.3. Controlul execuţiei unui program

Funcţia *echo* afişează liniile de program în timpul execuţiei acestora. În mod normal comenzile din fişierele-M nu sunt afişate în timpul execuţiei. Comanda de ecou

poate fi folosită pentru depanare sau pentru demonstrații.

3.6. Funcții de control a variabilelor din memorie

disp	Afișează o matrice sau un text;
clear	Șterge variabile și funcții;
pack	Compactează datele din memorie;
size	Returnează dimensiunea unei matrice;
length	Returnează lungimea unui vector.

3.6.1. Afișarea unui text

Funcția *disp* afișează o matrice fără să tipărească numele acesteia; se apelează cu sintaxa:

disp(X)

unde X este un șir text sau o matrice. De exemplu afișarea unei matrice cu etichete de coloane presupune secvența MATLAB:

```
disp(' Nr. Var1  Var2')
for i=1:4
      y=[i, 5*i^2, 2^i ];
      disp(y)
end
```

Se afișează rezultatul:

Nr.	Var1	Var2
1	5	2
2	20	4
3	45	8
4	80	16

3.6.2. Ștergerea variabilelor

Funcția *clear* șterge variabilele din spațiul de lucru al aplicației. Cu această funcție se poate șterge:

- o variabilă : `clear X`
- mai multe : `clear X Y Z`
- sau toate : `clear`

3.6.3. Compactarea memoriei

În timpul utilizării MATLAB-ului se creează și se șterg în permanență variabile din memoria calculatorului. Zona de memorie continuă, neocupată reducându-se, acest fapt conduce la imposibilitatea ca o variabilă mare să poată fi stocată, deși este mai mică decât memoria nefolosită. În această situație calculatorul va afișa mesajul „Out of memory". Compactarea spațiului de memorie rămas discontinuu se face cu funcția *pack;* se apelează cu sintaxa:

```
pack
```


3.6.4. Dimensiunea variabilelor

Pentru determinarea dimensiunilor variabilelor în MATLAB există două funcții: *size* și *length*.

Funcția *size* se apelează cu sintaxa:

[m, n]= size(X)

și returnează un vector linie cu două componente [m,n], care conțin numărul de linii m și numărul de coloane n, ale matricei X.

Funcția *length* se apelează cu sintaxa:

a=*length*(X)

și returnează dimensiunea maximă a matricei sau lungimea vectorului. Este echivalentă cu instrucțiunea *max*(*size*(X)).

Exemplul 3.6.1: Pentru matricea: X = [1 2 4; 5 7 9], instrucțiunile:

```
b = size(X)
[m,n] = size(X)
d = length(X)
```

au ca rezultat:

b = [2 3]	- 1 linie și 4 coloane
m = 2	- 1 linie
n = 3	- 4 coloane
d = 3	- dimensiunea maximă 4

3.7. Variabile speciale și constante în MATLAB

Variabilele speciale au un înțeles deosebit în MATLAB. Acestea nu pot fi declarate și sunt accesibile global în orice fișier-M. Variabilele speciale sunt introduse în mod obișnuit în construcțiile funcțiilor MATLAB, ele returnând valorile scalare utile. Variabilele și constantele speciale sunt:

ans - variabilă creată automat, în care este returnat rezultatul unui calcul, atunci când expresia nu a avut asignat un nume;

eps - variabilă permanentă în care este memorată eroarea relativă pentru calculele efectuate în virgulă mobilă. Valoarea implicită este eps = 2.2204e-016, dar poate fi redefinită la orice altă valoare;

pi - variabilă permanentă care are asignată valoarea 3.14159265358;

$i=\sqrt{-1}$ - variabilă folosită pentru introducerea numerelor complexe, ca în exemplul: z=2+3*i. Aceasta poate fi redescrisă și folosită ca o variabilă oarecare. Spre exemplu, dacă se definește s=sqrt(-1), variabila s poate fi folosită la introducerea numerelor complexe ca și variabilele i sau j, ca în exemplul: z=2+3*s;

$j=\sqrt{-1}$ - este o alternativă la utilizarea unității imaginare i;

inf - variabilă folosită pentru reprezentarea lui plus infinit în aritmetica IEEE, rezultat al împărțirii 1.0/0.0;

NaN - variabilă folosită pentru reprezentarea lui Not-a-Number (NaN), în aritmetica IEEE, rezultat al împărțirii nedefinite 0.0/0.0;

nargin - variabila permanentă pentru testarea numărului argumentelor de intrare ce trebuie introduse pentru apelarea unei funcţii;

nargout - variabilă permanentă pentru testarea numărului argumentelor de ieşire ale unei funcţii;

flops - returnează numărul de operaţii în virgulă mobilă efectuate de calculator. Adunările şi scăderile sunt contorizate fiecare câte o operaţie, dacă se efectuează între numere reale, şi două operaţii dacă se efectuează între valori complexe. Înmulţirile şi împărţirile sunt fiecare o operaţie dacă rezultatul este real şi 6 operaţii dacă rezultatul este complex. Aducerea la zero a contorului se realizează cu instrucţiunea *flops(0)*;

computer - variabilă folosită pentru obţinerea informaţiilor referitoare la tipul calculatorului şi numărul maxim de elemente pe care le poate gestiona versiunea respectivă de MATLAB. Cu instrucţiunea:

```
[calculator, elemente] = computer
```

se obţine:

calculator = PCWIN
elemente = 268435455

realmax - reprezintă cea mai mare valoare pozitivă în virgulă mobilă care poate fi folosită în calcule, respectiv 1.7977e+308;

realmin - reprezintă cea mai mică valoare pozitivă în virgulă mobilă care poate fi folosită în calcule, respectiv 2.2251e-308;

isieee - funcţie care returnează 1 dacă calculatorul lucrează cu arimetică IEEE şi 0 în caz contrar;

version, ver - funcţii pentru determinarea versiunii MATLAB şi a toolbox-urilor instalate pe calculator.

3.8. Importul şi exportul fişierelor de date

Funcţiile *load* şi *save* sunt comenzi MATLAB de încărcare şi salvare a fişierelor de pe (pe) discul curent:

load Încarcă variabilele dintr-un fişier de date de pe disc;

save Salvează variabilele într-un fişier de date pe disc.

Încărcarea unui fişier de date se face cu funcţia *load;* se apelează cu sintaxa:

load nume_fişier - format_date

unde „nume_fişier" este numele fişierului care se doreşte încărcat iar „format_date" este formatul datelor (ASCII sau BINAR) din acel fişier.

Funcţia *load* se utilizează pentru:

- încărcarea datelor din fişierul **matlab.mat**:
  load
- încărcarea datelor din fişiere în format **binar**, cu extensie ***.mat**:
  load nume_fişier
- încărcarea datelor din fişiere în format **binar**, cu extensie:
  load nume_fişier.extensie -mat
- încărcarea datelor din fişiere în format **ascii**, fără extensie:
  load nume_fişier -ascii

- încărcarea datelor din fişiere în format **ascii**, cu extensie:
  load nume_fişier.extensie -ascii

În toate cazurile, datele încărcate trebuie să fie tablouri complete. După încărcare datele se regăsesc în memoria calculatorului cu numele pe care l-a avut fişierul de date (fără extensie), indiferent de forma în care fuseseră stocate anterior. Spre exemplu, pentru încărcarea fişierului de date în format ASCII cu numele **temp.dat** se foloseşte instrucţiunea:

```
load temp.dat
```

Dacă fişierul nu are extensie, se foloseşte instrucţiunea:

```
load temp -ascii
```

În ambele cazuri, datele conţinute de fişierul încărcat se regăsesc în memorie cu numele **temp**.

Salvarea variabilelor din spaţiul de lucru foloseşte funcţia *save*; se apelează cu sintaxa:

save nume_fişier nume_variabile -format_date

unde „nume_fişier" este numele fişierului în care se salvează variabilele „nume_variabile", în formatul „format_date" (ASCII sau BINAR).

Funcţia *save* se utilizează pentru:

- salvarea în format **binar** a tuturor variabilelor din spaţiul de lucru, în fişierul **matlab.mat**:
  save
- salvarea în format **binar** a tuturor variabilelor din spaţiul de lucru, în fişierul **nume_fişier.mat**:
  save nume_fişier
- salvarea în format **binar** a a variabilelor A B C din spaţiul de lucru, în fişierul **nume_fişier.mat**:
  save nume_fişier A B C
- salvarea în format **ascii** (cu 8 digiţi) a variabilelor A B C din spaţiul de lucru, în fişierul **nume_fişier.ext**:
  save nume_fişier.ext A B C-ascii
- salvarea în format **ascii** (cu 16 digiţi) a variabilelor A B C din spaţiul de lucru, în fişierul **nume_fişier.ext**:
  save nume_fişier.ext A B C -ascii -double
- salvarea în format **ascii** (cu 16 digiţi) a variabilelor A B C din spaţiul de lucru, cu separarea datelor prin TAB-uri, în fişierul **nume_fişier.ext**:
  save nume_fişier.ext A B C -ascii -double -tabs

4. INTRODUCERE ÎN PROGRAMAREA MATLAB

4.1. Matrice, vectori şi scalari. Notaţii

MATLAB-ul este un pachet de programe care lucrează numai cu un singur tip de obiecte, matrice numerice rectangulare, cu elemente reale sau complexe. În acest sens, scalarii sunt asimilaţi matricelor cu o linie şi o coloană (1x1), iar vectorii sunt asimilaţi matricelor cu o linie (1xn) sau o coloană (nx1). Operaţiile şi comenzile în MATLAB sunt aproape naturale, în sens matriceal, asemănător modului de calcul obişnuit. Astfel entităţile:

$$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \quad B=\begin{bmatrix}0 & 1 & 5\\ 2 & 7 & 9\end{bmatrix} \quad C=\begin{bmatrix}1\\ 2\\ 3\end{bmatrix} \quad D=\begin{bmatrix}\sqrt{2} & 0 & \pi & 1\\ 3 & 2 & 1 & 5\\ 5 & 7 & 9 & 7\end{bmatrix}$$

$$E=[1\ 3\ 5\ 7]$$

sunt toate matrice în accepţiunea MATLAB. A este o matrice 2x2, B este 2x3, C este 3x1, D este 3x4 şi E=1x4.

Elementele unei matrice, fie aceasta A, pot fi identificate prin una dintre notaţiile: A_{ij}, A[i,j], A(i,j) etc. şi semnifică elementul de la intersecţia liniei i cu coloana j. Ultima notaţie, A(i,j), este cea care a fost adoptată şi în MATLAB şi va fi folosită în această lucrare. Dimensiunea unei matrice este precizată de o pereche de numere care arată numărul de linii şi coloane al matricei respective. (D este o matrice 3x4). O matrice cu o singură linie sau o singură coloană se numeşte vector linie (E) sau vector coloană (C), elementele acesteia putând fi identificate cu un singur indice. O matrice cu o singură linie şi o singură coloană este un scalar.

Pentru a face referire la un element A(i,j) al unei matrice A, sunt necesari doi indici, indicele de linie şi indicele de coloană, în această ordine. Referirea unui element al unui vector poate fi făcută numai cu un singur indice. Astfel:

A(2,1) = 3; B(1,3) = 5, C(2) = 2; D(3,4) = 7; E(4) = 7.

4.2. Definirea matricelor simple

Definirea matricelor se face prin una dintre următoarele metode:
- introducerea explicită a listei de elemente;
- generarea prin instrucţiuni şi funcţii;
- crearea de fişiere M;
- încărcarea din fişiere de date externe.

MATLAB-ul nu conţine instrucţiuni de dimensionare şi declaraţii de tip, iar memoria este alocată automat, până la valoarea maxim disponibilă. Cea mai simplă metodă de definire a matricelor mici constă în utilizarea unei liste explicite. La introducerea unei astfel de liste trebuie respectate următoarele reguli:
- elementele unei linii trebuie separate prin blanc-uri sau virgule;
- liniile se separă prin semnul punct-virgulă „ ; " ;
- elementele matricei sunt cuprinse între paranteze drepte „ [ ] ".

Astfel, matricea A din §4.1 poate fi introdusă cu secvenţa:

```
A=[1 2; 3 4]
```

care returnează rezultatul:

```
A=
     1     2
     3     4
```

Matricea A, astfel definită, poate fi utilizată în calcule sau poate fi salvată într-un fişier de date pentru o folosire ulterioară.

Pentru matricele mari, la care datele de intrare nu încap pe o singură linie, se poate proceda la înlocuirea semnului „ ; " cu „Enter", ca în exemplul următor:

```
A=[   1     2
      3     4]
```

Un alt mod de definire a unei matrice constă în apelarea numelui unui fişier de date aflat pe disc. Acest fişier trebuie să fie înregistrat în format ASCII (text), organizat ca o matrice rectangulară (completă) şi să aibă extensia *.m. Astfel, dacă pe disc se găseşte un fişier cu numele **date.m** care conţine următoarele două linii de text:

```
1     2
3     4
```

instrucţiunea care citeşte fişierul şi generează matricea A este:

```
A=date
```

Posibilitatea de a importa date de la alte programe este prezentată în §3.8.

4.3. Elementele matricelor

Elementele matricelor pot fi numere reale sau complexe, precum şi orice expresie MATLAB. De exemplu, pentru:

```
x=[-1.3 sqrt(3) (1+2+3)*4/5]
```

rezultă:

```
x = [ -1.3000   1.7321   4.8000 ]
```

Elementele unei matrice pot fi referite cu indici cuprinşi între paranteze rotunde „( )", ca în exemplul:

```
a = x(2)
```

care returnează:

```
a =1.7321
```

De remarcat că dacă se asignează o valoare unui element care ocupă o poziţie în afara dimensiunii maxime a matricei sau vectorului referit, dimensiunea acestuia este mărită automat până la valoarea indicelui noului element, iar elementele nedefinite sunt setate la valoarea zero. În acest sens, instrucţiunea:

```
x(5)= abs(x(1))
```

returnează rezultatul:

```
x = [ -1.3000   1.7321   4.8000        0   1.3000 ]
```

iar instrucţiunea:

```
A(2,4)= 6
```

returnează rezultatul:

```
A =
   1   2   0   0
   3   4   0   6
```

O modalitate de a construi matrice mari constă în folosirea matricelor mici ca elemente. Spre exemplu, din două matrice 2x3 se poate construi o matrice 4x3; dimensiunile matricelor utilizate trebuie să fie astfel alese încât să realizeze tablouri rectangulare complete.

Fie A1 şi A2 cele două matrice utilizate. Cu secvenţa următoare:

```
A1 = [1 2; 3 4]
A2 = [5 6; 7 8]
A  = [A1; A2]
```

se obţine rezultatul:

```
A =
   1   2
   3   4
   5   6
   7   8
```

O matrice mai mică poate fi extrasă din matrice mai mari utilizând semnul „:" (două puncte). De exemplu:

```
B = A(2:3,:)
```

extrage liniile doi şi trei şi toate coloanele din matricea curentă A, obţinându-se matricea B:

```
B =
   3   4
   5   6
```

4.4. Declaraţii şi variabile

MATLAB-ul este un limbaj de expresii. Expresiile tipărite de utilizator sunt interpretate şi evaluate. Instrucţiunile MATLAB sunt, de cele mai multe ori, de forma:

variabila = expresie

sau, mai simplu:

expresie

Expresiile sunt compuse din operatori sau alte caractere speciale, din funcţii şi nume de variabile. Evaluarea expresiei produce o matrice, care este afişată pe ecran şi atribuită unei variabile. Dacă numele variabilei şi semnul egal („variabila=") sunt omise, MATLAB-ul crează automat o variabilă cu numele „ans", ca în exemplul:

```
3/4
```

care returnează:

ans = 0.7500

Orice instrucţiune este în mod normal terminată cu „Enter". Dacă ultimul caracter al acesteia este punct-virgulă „ ; ", instrucţiunea este executată, dar tipărirea este suprimată. Utilizarea acestui caracter la sfârşitul unei instrucţiuni în fişiere-M este necesară în situaţiile în care nu se doreşte afişarea datelor intermediare. De exemplu:

```
A=[1 2 3; 4 5 6;7 8 9];
```

introduce matricea A, dar nu o afişează. Tastarea numelui unei variabile urmată de „Enter" afişează valoarea acesteia.

Dacă expresia este aşa de mare încât declaraţia nu încape pe o singură linie, se

utilizează semnul „..." (trei puncte), urmat de „Enter", pentru a preciza că instrucţiunea continuă pe linia următoare. Astfel, instrucţiunea:

```
S=1+2+3+...
  4+5+6;
```

evaluează suma celor şase numere şi o atribuie variabilei S. Spaţiile dintre semnele „=", „+", „-" şi numere sunt opţionale.

Numele de variabile şi funcţii au ca prim caracter o literă, urmată de litere, cifre sau caracterul „liniuţă de subliniere" (adică „ _ "). Deşi se pot folosi oricâte caractere, MATLAB-ul reţine ca nume de variabilă numai primele 19 caractere. MATLAB-ul face deosebirea între literele mari şi mici, asfel încât „a" şi „A" sunt două variabile dinstincte. Funcţia *„casesen"* permite trecerea MATLAB-ului în modul senzitiv/nesenzitiv, în vederea separării literelor mari de cele mici. Numele de funcţii este obligatoriu să fie redate cu litere mici. MATLAB-ul este la lansare în modul senzitiv, adică face dinstincţie între literele mari şi cele mici, dar cu comanda:

```
casesen off
```

trece în modul nesenzitiv; revenirea se face cu comanda:

```
casesen on
```

4.5. Numere şi expresii aritmetice

MATLAB-ul utilizează pentru reprezentarea numerelor notaţia zecimală convenţională, cu punct zecimal. Puterea este un factor de scală inclus ca sufix, ca şi în notaţia ştiinţifică. Exemple de reprezentări de numere în MATLAB, sunt:

3	-99	0.0001
9.6397238	1.60210E-20	6.02252e23

Calculatoarele care utilizează aritmetica în virgulă mobilă IEEE au precizia relativă a numerelor „eps", aproximată cu 16 digiti semnificativi. Limitele numerelor folosite în MATLAB sunt: 10^{-308} şi 10^{308}.

MATLAB-ul utilizează următorii operatori aritmetici:

+	adunarea
-	scăderea
*	înmulţirea
/	împărţirea la dreapta
\	împărţirea la stânga
^	ridicarea la putere

Există două simboluri pentru împărţire, utilizate în mod special pentru operaţiile cu matrice. Pentru expresiile scalare 1/4 şi 4\1 se obţine acelaşi rezultat (0.25). Într-o expresie, parantezele sunt utilizate pentru a preciza ordinea executării operaţiilor.

4.6. Structura programelor MATLAB

MATLAB-ul lucrează fie în modul linie de comandă, situaţie în care fiecare linie este prelucrată imediat şi rezultatele sunt afişate, fie cu programe conţinute în fişiere. Aceste două moduri formează împreună un „mediu" de programare.

Fişierele ce conţin instrucţiuni MATLAB se numesc fişiere-M (deoarece au extensia „.m") şi sunt programe MATLAB. Un fişier-M constă dintr-o succesiune de instrucţiuni MATLAB, cu posibilitatea apelării altor fişiere-M precum şi a apelării recursive.

Un program MATLAB poate fi scris sub forma fişierelor „script" sau a fişierelor „function". Ambele tipuri de fişiere sunt scrise în format ASCII, iar algoritmul care a fost implementat poate fi urmărit cu foarte multă uşurinţă, dacă se cunosc convenţiile şi sintaxa MATLAB. Aceste tipuri de fişiere, obligatoriu cu extensia *.m, permit crearea unor funcţii noi care le pot completa pe cele deja existente. Prin această facilitate, MATLAB-ul poate fi extins la aplicaţii specifice utilizatorului, care are posibilitatea să scrie noi proceduri.

4.6.1. Fişierele script

Un fişier *„script"* este un fişier extern care conţine o secvenţă de comenzi MATLAB. Prin apelarea numelui fişierului, se execută secvenţa MATLAB conţinută în acesta. După execuţia completă a unui fişier script, variabilele cu care acesta a operat rămân în zona de memorie a aplicaţiei. Aceste fişiere nu permit integrarea în programe mari, realizate pe principiul modularizării. Fişierele script sunt folosite pentru rezolvarea unor probleme care cer comenzi succesive atât de lungi, încât ar putea deveni greoaie pentru lucrul în mod interactiv, adică în modul linie de comandă.

4.6.2. Fişierele funcţie

Dacă prima linie a fişierului-M conţine cuvântul *„function"*, fişierul respectiv este declarat ca fişier funcţie. O funcţie diferă de un *„script"* prin faptul că poate lucra cu argumente. Variabilele definite şi manipulate în interiorul fişierului funcţie sunt localizate la nivelul acesteia. Prin urmare, la terminarea execuţiei unei funcţii, în memoria calculatorului nu rămân decât variabilele de ieşire ale acesteia.

Fişierele funcţie sunt utilizate pentru extinderea MATLAB-ului, adică pentru crearea unor noi funcţii MATLAB. Forma generală a primei linii a unui fişier funcţie este:

function [param_ieşire]= nume_funcţie(param_intrare)

unde:

function - este cuvânt cheie care declară fişierul ca fişier funcţie (obligatoriu);

nume_funcţie - numele funcţiei, adică numele sub care se salvează fişierul, fără extensie. Nu poate fi identic cu cel al unui fişier-M preexistent.

param_ieşire - parametri de ieşire trebuie separaţi cu virgulă şi cuprinşi între paranteze drepte. Dacă funcţia nu are parametri de ieşire, parantezele drepte şi semnul egal nu mai au sens.

param_intrare - parametri de intrare trebuie separaţi cu virgulă şi cuprinşi între paranteze rotunde. Dacă funcţia nu are parametri de intrare, parantezele rotunde nu mai au sens.

Aceste fişiere pot fi adăugate ca funcţii noi în structura MATLAB. Comenzile şi funcţiile care sunt utilizate de noua funcţie sunt înregistrate într-un fişier cu extensia .m. Prima linie a fişierului trebuie să conţină definiţia sintactică a noii funcţii. De exemplu, un fişier funcţie numit **„medie.m"**, care calculează media aritmetică, poate avea următoarea formă :


```
function m = medie(x)
n=length(x);
m = sum(x)/n;
y=['Media numerelor este:',num2str(m)];
disp(y)
```

Exemplul 4.6.1. Să se scrie un fişier „function" care să calculeze transformata Fourier a unui semnal şi să returneze vectorii frecvenţă şi amplitudine. Reprezentaţi grafic amplitudinea funcţie de frecvenţă.

Se selectează succesiv **File/New/M-file** şi se scriu următoarele instrucţiuni:

```
function [f,X]= test4_1
t=0:.001:.039;
fe=1/(t(2)-t(1))
x=2*sin(2*pi*50*t)+1.5*sin(2*pi*100*t)+...
sin(2*pi*200*t)+.5*sin(2*pi*350*t)
Xt=fft(x); Xm=abs(Xt);
N=length(x);X=Xm(1:N/2+1); f=[0:N/2]*fe/N;
subplot(211); plot(t,x); subplot(212); stem(f,X)
```

Se înregistrează acest fişier cu numele **test4_1.m**, selectând succesiv **File/Save** şi completând numele menţionat în caseta de dialog. Apelând numele **„test4_1"** (fără extensie), se lansează în execuţie fişierul funcţie **„test4_1.m"**, care afişează semnalul temporal în semi-ecranul superior şi spectrul de frecvenţă în semi-ecranul inferior, figura 4.1.

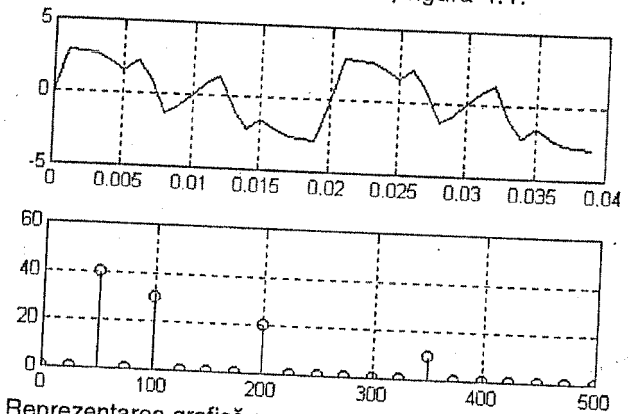


Fig. 4.1. Reprezentarea grafică a componentelor spectrale ale unui semnal.

Acest exemplu simplu poate constitui baza de pornire pentru construirea unui analizor spectral software. Dacă prima linie a programului din exemplul 4.6.1 este omisă, programul devine un fişier script, care nu poate fi apelat cu parametri formali sau integrat într-un program modularizat.

4.7. Comentariile şi help-ul într-un program

Un comentariu se introduce într-un program MATLAB prin caracterul procent, (%), plasat la începutul liniei. Dacă semnul procent (%) apare pe prima poziţie într-o linie, aceasta va fi omisă de compilator, iar dacă apare într-o linie de program, partea de linie care urmează va fi omisă. Într-un program, un comentariu poate apărea în orice poziţie; este recomandată însă prezenţa unui comentariu după prima linie care declară o funcţie. În acest caz, comentariul, care apare imediat după linia de declarare a funcţiei, constituie **help**-ul fişierului respectiv (cunoscut sub numele de **„grupul de linii H"**); dacă se apelează:

help nume_funcţie

se afişează liniile care încep cu semnul procent şi sunt situate între prima linie după declararea funcţiei şi prima linie liberă sau prima instrucţiune ce urmează acestui grup compact de linii.

Prin urmare, pentru a ataşa un HELP unui program MATLAB, se scriu liniile acestuia începând cu semnul procent „%". După ultima linie a help-ului se lasă o linie liberă sau se trece direct la o instrucţiune MATLAB, ca în exemplul:

function nume_funcţie (această linie lipseşte la fişierele **script**)
% Prima linie Help
% A doua linie Help
%.....
% Ultima linie Help
linie liberă (este oligatorie dacă urmează un alt comentariu care nu se doreşte a fi linie Help şi opţională în caz contrar)
Instrucţiuni MATLAB

Dacă un fişier *.m are structura prezentată mai sus şi se apelează:

help nume_funcţie

se va afişa mesajul:

Prima linie Help
A doua linie Help
.....
Ultima linie Help

Exemplul 4.7.1. Să se scrie un program prevăzut cu *help*.
Se selectează succesiv **File/New/M-file** şi se scriu următoarele instrucţiuni:

```
function test4_2
%Acest fisier reprezinta un semnal discret.
t=0:0.001:.059;
x=sin(2*pi*100*t)
stem(t,x)
```

Se înregistrează acest fişier cu numele **test4_2.m**, selectând succesiv **File/Save** şi completând numele menţionat în caseta de dialog. Dacă se apelează:

```
help test4_2
```

se obţine pe ecran mesajul:

```
Acest fisier reprezinta un semnal discret.
```


4.8. Un exemplu simplu de program complet

Exemplul 4.8.1. Să se completeze programul prezentat în §4.6 cu: scala de etalonare în amplitudine, *help* şi comentarii.
Se selectează succesiv **File/New/M-file** şi se scriu următoarele instrucţiuni:

```
function [f,X]= test4_3
% Acest program realizeaza analiza spectrala
% a unui semnal x, cu trei componente.
t=0:.001:.099; % momentele de esantionare
fe=1000; % frecventa de esantionare
x=4*sin(2*pi*50*t)+2*sin(2*pi*100*t)+...
6*sin(2*pi*350*t); %semnalul temporal
Xt=fft(x); % transformata Fourier
N=length(x); % lungimea secventei
Xm=abs(Xt); % determina modulul
X=Xm(1,1:N/2+1)/(N/2);
f=[0:N/2]*fe/N; % frecventele pozitive
subplot(211)  %selecteaza fereatra sup.
plot(t,x)  % reprezinta semnalul continuu
grid % traseaza retea grid
xlabel('t[s]')  %eticheteaza axa x
ylabel('x(t)[V]') % eticheteaza axa y
title('Semnalul temporal') % pune titlul
subplot(212) %selecteaza fereatra inf.
stem(f,X); %reprezinta valori discrete
xlabel('f[Hz]')
ylabel('X(f)[V]')
grid
title('Analiza spectrala a unui semnal')
```

Se înregistrează acest fişier cu numele **test4_3.m**, selectând succesiv **File/Save** şi completând numele menţionat în caseta de dialog. Rezultatele vor fi afişate grafic, sub forma prezentată în figura 4.2. Se observă că au fost determinate toate cele trei componente spectrale conţinute de semnalul temporal x. De fapt, acesta constituie un modul de analizor spectral, care poate fi utilizat ori de câte ori un semnal temporal, simulat sau achiziţionat cu ajutorul unor mijloace specializate, trebuie descompus în componentele sale armonice

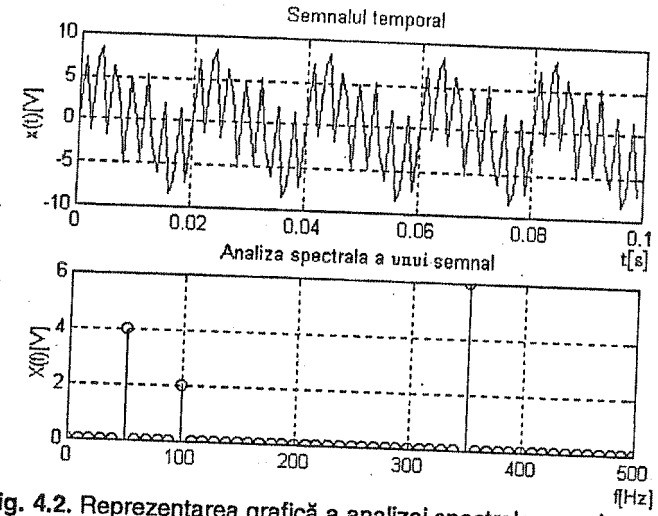


Fig. 4.2. Reprezentarea grafică a analizei spectrale a unui semnal

5. INSTRUCŢIUNI ŞI FUNCŢII DE CONTROL

5.1. Instrucţiuni de control logic

Instrucţiunile de control logic în MATLAB sunt:

if	Instrucţiune pentru execuţia condiţionată;
else	Instrucţiune asociată cu „if";
elseif	Instrucţiune asociată cu „if";
for	Instrucţiune pentru crearea ciclurilor cu număr specificat de paşi;
while	Instrucţiune pentru crearea ciclurilor cu condiţie logică;
break	Instrucţiune pentru terminarea forţată într-un ciclu;
return	Returnează execuţia la funcţia precizată;
error	Instrucţiune pentru afişarea unui mesaj de eroare;
end	Instrucţiune pentru încheierea ciclurilor „for", „while" şi „if".

Programele MATLAB sunt scrise pentru a realiza numai paşi secvenţiali, operaţiile fiind realizate una după alta. În multe cazuri, însă, este necesară repetarea unui set de instrucţiuni atâta timp cât o condiţie este realizată şi a altui set de instrucţiuni, dacă condiţia nu mai este realizată. Alteori, este necesară repetarea unui grup de instrucţiuni de un anumit număr de ori. MATLAB pune la dispoziţia programatorului două instrucţiuni, *for* şi *while*, cu ajutorul cărora un grup de instrucţiuni, numit corpul ciclului, se poate repeta de mai multe ori.

Instrucţiunea *for* este folosită pentru repetarea unui grup de instrucţiuni de un anumit număr de ori, cunoscut iniţial. Un contor de ciclu ţine evidenţa numărului de repetări ale instrucţiunilor din corpul ciclului şi opreşte ciclul când contorul ajunge la valoarea precizată.

Instrucţiunea *while* este folosită atunci când calculele efectuate cu un grup de instrucţiuni trebuie repetate până când o anumită condiţie este îndeplinită. Numărul de iteraţii nu este cunoscut iniţial. Ieşirea din ciclu se realizează la îndeplinirea condiţiei impuse, care este testată la începutul fiecărei iteraţii.

5.1.1. Instrucţiunea condiţională „if"

În cadrul unui algoritm este deseori necesară o selecţie a grupului de instrucţiuni ce urmează a fi executate, condiţionată de valoarea de adevăr a unei expresii. Instrucţiunile condiţionale utilizează operatorii relaţionali şi operatorii logici.

5.1.1.1. Operatorii relaţionali

MATLAB-ul are şase operatori relaţionali, care sunt utilizaţi pentru a compara două matrice de dimensiuni egale; lista este prezentată în tabelul 5.1.

Operatorii relaţionali compară două matrice sau două expresii matriceale, element cu element. Aceştia returnează o matrice de aceeaşi dimensiune cu a matricelor care se

compară, cu elementele 1 când relaţia este ADEVARATĂ şi cu elementele zero când relaţia este FALSĂ. Primii patru operatori (< ,<= , >, >=) compară numai partea reală a operanzilor (partea imaginară este ignorată), iar ultimii doi operatori (== şi ~=) tratează atât partea reală cât şi cea imaginară.

Tabelul 5.1. Operatorii relaţionali

Operatorii relaţionali	Semnificaţia
<	mai mic
<=	mai mic sau egal
>	mai mare
>=	mai mare sau egal
= =	identic
~ =	diferit

Forma generală de utilizare a operatorilor relaţionali este :

rezultat = expresie_1 op_relaţional expresie_2

unde:

rezultat - este o matrice de elemente 0 şi 1, în care sunt returnate rezultatele comparaţiei;

expresie_1, expresie_2 - matricele sau expresiile matriceale care se compară ;

op_relaţional - unul dintre cei şase operatori relaţionali menţionaţi în tabelul 5.1.

Dacă A şi B sunt matricele care se compară şi C matricea în care este returnat rezultatul, expresia:

C=A *op_relaţional* B

returnează:

$$C(i,j) = \begin{cases} 1 & \text{,dacă expresia (A(i,j) op\_relaÆional B(i,j)) este ADEVARATĂ} \\ 0 & \text{,dacă expresia (A(i,j) op\_relaÆional B(i,j)) este FALSĂ} \end{cases}$$

Dacă unul dintre operanzi este un scalar şi celălalt o matrice, scalarul se „extinde" până la dimensiunea matricei.

Exemplul 5.1.1. Următoarele două instrucţiuni:

```
X=5>=[1 2 3;4 5 6;7 8 10]
X=5*ones(3,3)>=[1 2 3;4 5 6;7 8 10]
```

sunt echivalente, obţinându-se acelaşi rezultat:

```
X=
     1     1     1
     1     1     0
     0     0     0
```

Exemplul 5.1.2. Fie matricele :

$$A = \begin{bmatrix} 1 & -i & 0 \\ 2-i & 0 & 1+i \end{bmatrix} \qquad B = \begin{bmatrix} 1 & -i & i \\ 2-i & 2 & 1-5i \end{bmatrix}$$

Să se realizeze comparaţiile:

C=A>=B

D=A==B

Cu secvenţa MATLAB:

```
A=[1 -i 0;2-i 0 1+i];
B=[1 -i i;2-i 2 1-5*i];
C=A>=B
D=A==B
```

Se obţin rezultatele:

```
C =                    D =
   1   1   1              1   1   0
   1   0   1              1   0   0
```

5.1.1.2. Operatorii logici

Pentru combinarea a două sau mai multe expresii logice se utilizează operatorii logici din tabelul 5.2.

Tabelul 5.2. Operatorii logici

Operatori logici	Simbol MATLAB	Prioritatea
NU	~	1
ŞI	&	2
SAU	|	3

Operatorii & şi | compară doi scalari sau două matrice de dimensiuni egale. Pentru matrice operează element cu element. Operatorii logici returnează „1" (ADEVĂRAT) pentru orice valoare diferită de zero. Se va returna „1" pentru ADEVĂRAT şi „0" pentru FALS.

Operatorul logic NU (sau complementul logic), este operator unar. Expresia ~A returnează „0" dacă A e diferit de zero şi „1" dacă A este „0".

Operatorii logici au prioritate mai mică decât operatorii relaţionali şi aritmetici. Astfel, se evaluează întâi expresiile care conţin operatori relaţionali şi aritmetici, şi apoi cele cu operatori logici.

Operatorii logici se utilizează întotdeauna pentru compararea matricelor cu elemente 0 şi 1, calculate cu operatori relaţionali. Spre exemplu, expresia logică:

$$D(i,j) = \begin{cases} 1 & \text{,dacă } (A(i,j) < B(i,j)\ \&\ (B(i,j) < C(i,j)) \\ 0 & \text{,î n rest} \end{cases}$$

are următoarea descriere în MATLAB:

D = (A < B) & (B < C)

Această operaţie este posibilă numai dacă matricele rezultate din operaţiile A< B şi B < C au aceeaşi dimensiune.

Operatorii logici se utilizează cu expresii logice complete. Spre exemplu, expresia a<b & b<c este corectă, însă expresia a<b &c este incompletă. Expresiile logice pot fi precedate de operatorul logic NU. Acest operator schimbă valorile expresiei în valorile opuse: dacă a>b este 1 (ADEVĂRAT), atunci ~(a>b) este 0 (FALS). Expresiile logice pot conţine mai mulţi operatori logici, ca în exemplul:

~(b= =c | c = = 5)

Ordinea de prioritate a operatorilor logici, de la superior la inferior, este: NU, ŞI, SAU. Dacă b = 3 şi c = 5, în exemplul anterior, prima expresie este 0 şi a doua 1. Se evaluează apoi expresia 0 | 1 a cărei valoare logică este 1. Aplicarea operatorului NU

face ca rezultatul să fie 0. Sintaxa MATLAB asociată exemplului este:

```
b = 3; c = 5;
d = ~ (b == c | c == 5)
```

Lipsa parantezelor în expresia de mai sus nu lasă nemodificată valoarea acesteia. Astfel :

```
b = 3 ; c = 5
d = ~b == c | b == 5
```

returnează valoarea logică d = 0 (FALS), în timp ce:

```
b = 3 ; c = 5
d = ~(b == c | b == 5)
```

are valoarea logică d = 1.

Trebuie reţinut că în MATLAB, orice valoare diferită de zero este considerată adevărată logic, adică 1. Pentru ca cititorul să se familiarizeze cu utilizarea operatorilor logici şi relaţionali se recomandă descompunerea expresiilor complexe în paşi care pot fi urmăriţi cu uşurinţă.

În tabelul 5.3 sunt prezentate toate combinaţiile posibile dintre expresiile logice A şi B şi operatorii logici: ŞI, SAU şi NU.

Tabelul 5.3. Tabela de adevăr a operatorilor logici

A	B	~A	A | B	A&B
FALS	FALS	FALS	FALS	FALS
FALS	ADEV.	ADEV.	ADEV.	FALS
ADEV.	FALS	FALS	ADEV.	FALS
ADEV.	ADEV.	FALS	ADEV.	ADEV.

Exemplul 5.1.3. Următoarele două expresii scalare ilustrează prioritatea operatorilor relaţionali:

$$1 \,\&\, 0 + 3$$
$$3 > 4 \,\&\, 1$$

Prima expresie are valoarea logică 1 deoarece 0+3=1 şi 1&3=1, iar a doua are valoarea logică 0 deoarece (3>4)=0 şi 0&1=0.

Exemplul 5.1.4. Să se descrie şi să se reprezinte grafic funcţia modul, definită de relaţia:

$$t = \begin{cases} -t, & \text{dacă } -4 \le t < 0 \\ t, & \text{dacă } 0 \le t \le 4 \end{cases}$$

Secvenţa de program este:

```
a=-4; b=4; d=.01; c=0;
t=a:d:b;
k=max(size(t));
for i=1:k
      if (t(i)>=a)&(t(i)<c)
            y(i)=abs(t(i));
      end
      if (t(i)>=c)&(t(i)<=b)
            y(i)=t(i);
```



```
            end
      end
      plot(t,y)
      title('Functia modul')
```

Exemplul 5.1.5. Să se descrie şi să se reprezinte grafic funcţia definită de relaţia:

$$y(x) = \begin{cases} \sin(t) & \text{, dacă } \{a \le t < b\} \cup \{c \le t \le d\} \\ \cos(5t) & \text{, dacă } b \le t < c \end{cases}$$

Secvenţa de program este:

```
a=-10; b=-4; c=4; d=10;
f=.01;   % pasul vectorizarii
t=a:f:d;
k=max(size(t));
for i=1:k
   if (((t(i)>=a)&(t(i)<b))|((t(i)>=c)&(t(i)<=d)))
         y(i)=sin(t(i));
   end
   if ((t(i)>=b)&(t(i)<c))
         y(i)=cos(5.*t(i));
   end
end
plot(t,y)
```

Graficul rezultat este reprezentat în figura 5.1.

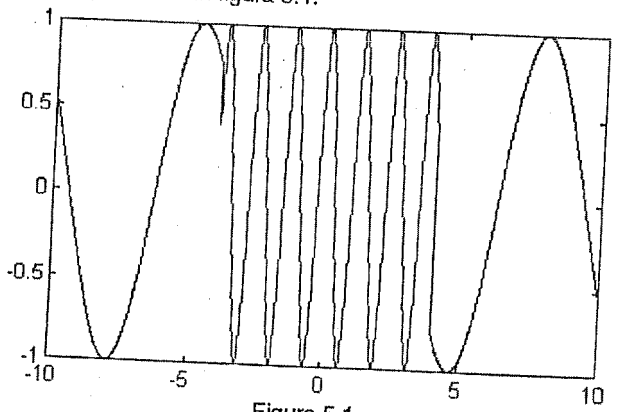


Figura 5.1.

5.1.1.3. Instrucţiunea „if" simplă

Instrucţiunea *„if"* poate fi implementată ca instrucţiune *„if"* simplă, sau poate include clauzele *else* sau *elseif*. Forma generală a unei instrucţiuni *„if"* simplă, este următoarea:

if expresie_ logică
  grup_de_instrucţiuni
end

Dacă expresia_logică este adevărată, se execută grupul de instrucţiuni dintre instrucţiunea *if* şi instrucţiunea *end*. Dacă expresia logică este falsă, se trece la prima instrucţiune care urmează după instrucţiunea *end*.

Pentru o citire mai uşoară, cât şi pentru o urmărire a grupurilor de instrucţiuni care se execută, se procedează la indentarea (deplasarea) acestora.

Fie următoarea secvenţă MATLAB:

```
if a < 50
   k=k+1
   s=s+a
end
```

Dacă a este un scalar şi dacă a< 50, k se incrementează cu 1 şi apoi a este adunat cu s; altfel, cele două instrucţiuni sunt omise, s rămânând cu valoarea cu care a fost iniţializat înainte de a ajunge la instrucţiunea *if*. Dacă a este un vector sau o matrice, atunci k este incrementat cu 1 şi a este însumat cu s, numai dacă fiecare element al lui a este mai mic decât 50.

Instrucţiunea *if* poate fi inclusă în interiorul unei alte instrucţiuni *if*, ca în exemplul următor:

```
if expresia_logică_1
        grupul_de_instrucţiuni_A
if expresia_logică_2
        grupul_de_instrucţiuni_B
end
        grupul_de_instrucţiuni_C
end
        grupul_de_instrucţiuni_D
```

Dacă expresia_logică_1 este 1 (ADEVĂRATĂ), se execută întotdeauna grupurile de instrucţiuni A şi C. Dacă şi expresia_logică_2 este 1 (ADEVĂRATĂ), se execută grupul de instrucţiuni B, înaintea executării grupului de instrucţiuni C. Dacă expresia_logică_1 este 0 (FALSĂ), se trece la grupul de instrucţiuni D. Indentarea este o operaţie necesară la utilizarea instrucţiunilor *if*, deoarece pune în evidenţă grupurile de instrucţiuni care se execută funcţie de condiţiile pe care le îndeplinesc.

Fie următorul exemplu de utilizare a instrucţiunilor condiţionale *„if"* succesive:

```
if a < 50
      k = k+1
      s=s+a
      if b > a
            b=0
      end
end
```

Dacă a şi b sunt scalari şi dacă a < 50, se incrementează k cu 1 şi se adună a la s. În schimb, dacă b > a, se setează b la zero. Dacă a nu este mai mic decât 50, se trece la grupul de instrucţiuni care urmează după a doua instrucţiune *end*.

Dacă a este un vector sau o matrice, condiţia a < 50 este adevărată numai

dacă fiecare element al lui a este mai mic decât 50. Dacă a şi b sunt vectori sau matrice, atunci b > a numai dacă fiecare pereche (b, a) verifică această condiţie.

Dacă a sau b este scalar, comparaţia se face între fiecare element al matricei cu scalarul extins la o matrice.

5.1.1.4. Clauza „else"

Clauza *else* este utilizată pentru a executa un set de instrucţiuni, dacă expresia logică este ADEVĂRATĂ şi un alt set de instrucţiuni, dacă expresia logică este FALSĂ. Forma generală a instrucţiunii *if* este combinată cu clauza *else* (numită uneori instrucţiunea *if-else*) ca în exemplul următor:

```
if expresie_logică
        grupul_de instrucţiuni_A
else
        grupul_de_instrucţiuni_B
end
```

Dacă expresia logică este adevărată se execută grupul de instrucţiuni A, iar dacă este falsă se execută grupul de instrucţiuni B.

Exemplul 5.1.6. Pentru următoarea funcţie:

$$f(x) = \begin{cases} 2x+8 & , \text{ dacă } x \le 2 \\ 3x^2 & , \text{ dacă } x > 2 \end{cases}$$

descrierea MATLAB este:

```
if x ≤ = 2
        f = 2*x+8
else
        f=3*x^2
end
```

5.1.1.5. Clauza „elseif"

Dacă funcţia de calculat are mai multe nivele de instrucţiuni *if-else*, este dificilă determinarea expresiei logice adevărate, care selectează grupul de instrucţiuni ce urmează a fi executat. În acest caz, se utilizează clauza *elseif*:

```
if expresia_logică_1
        grupul_de_instrucţiuni_A
elseif expresia_logică_2
        grupul_de_instrucţiuni_B
elseif expresia_logică_3
        grupul_de_instrucţiuni_C
end
```

- dacă expresia_logică_1 este adevărată, este executat numai grupul de instrucţiuni A;
- dacă expresia_logică_1 este falsă şi expresia_logică_2 este adevărată, se execută numai grupul_de_instrucţiuni_B;
- dacă expresiile logice 1 şi 2 sunt false, iar expresia logică 3 este adevărată, se

execută numai grupul de instrucţiuni C;

- dacă mai multe expresii logice sunt adevărate, prima instrucţiune logică adevărată determină care grup de instrucţiuni este executat prima dată;
- dacă nici o expresie logică nu este adevărată, nu se execută nici un grup de instrucţiuni din structura *if*.

Clauza *elseif* poate fi combinată cu clauza *else* într-o structură generală de forma:

```
if expresia_logică-1
   grupul_de_instrucţiuni_A
elseif expresia_logică_2
   grupul_de_instrucţiuni_B
elseif expresia_logică_3
   grupul_de_instrucţiuni_C
else
   grupul_de_instrucţiuni_D
end
```

Dacă nici o expresie logică dintre primele trei nu este adevărată, se va executa grupul_de_instrucţiuni_D.

Exemplul 5.1.7. Să se genereze o matrice A, cu n linii şi n+1 coloane, ale cărei elemente sunt:

$$A=\begin{cases} 2 & \text{, dacă } i=j \\ -1 & \text{, dacă } |i-j|=1 \\ 0 & \text{, în rest} \end{cases}$$

Cu secvenţa MATLAB:

```
n=4;
for i=1:n,
for j=1:n+1,
      if  i==j,
                A(i,j)=2
            elseif abs(i-j)==1,
                A(i,j)=-1;
            else
            A(i,j)=0;
            end
      end
 end
A
```

se obţine rezultatul:

```
A =
   2  -1   0   0   0
  -1   2  -1   0   0
   0  -1   2  -1   0
   0   0  -1   2  -1
```


5.1.2. Instrucţiunea repetitivă „for"

Instrucţiunea *for* permite repetarea unui grup de instrucţiuni din corpul buclei, de un anumit număr de ori; are următoarea structură generală:

```
for index = expresie
   grupul_de_instrucţiuni
end
```

unde:

- index este numele contorului;
- expresie este o matrice, un vector sau un scalar;
- grup_de_instrucţiuni este orice expresie MATLAB.

În aplicaţii, „expresie" este de cele mai multe ori de forma:

k = iniţial:pas:final

unde:

iniţial - este prima valoare a lui k,
pas - pasul (dacă este omis, este considerat 1),
final - cea mai mare valoare pe care o poate lua k.

La fiecare pas de calcul „index" are valoarea unuia dintre elementele expresiei. Dacă „expresie" este o matrice, ciclarea se face pe coloane.

Pentru un ciclu *for* cu pasul negativ sau neîntreg se generează mai întâi un vector cu pasul şi limitele dorite şi apoi se citesc valorile acestuia în cadrul buclei *for*.

La folosirea buclei *for* trebuie respectate următoarele reguli:

1. indexul buclei *for* trebuie să fie o variabilă;
2. dacă expresia este o matrice goală, bucla nu se execută. Se va trece la următoarea instrucţiune după instrucţiunea *end*;
3. dacă expresia este un scalar, bucla se execută o singură dată, cu indexul dat de valoarea scalarului;
4. dacă expresia este un vector linie, bucla se execută de atâtea ori câte elemente are vectorul, de fiecare dată indexul având valoarea egală cu următorul element din vector;
5. dacă expresia este o matrice, indexul va avea la fiecare iteraţie valorile conţinute în următoarea coloană a matricei;
6. la terminarea ciclurilor *for*, indexul are ultima valoare utilizată;
7. dacă se utilizează operatorul două puncte („:") pentru a defini expresia, ca în exemplul:

for k = iniţial:pas:final

bucla se execută de:

$$n=\left[\frac{\text{final - initial}}{\text{pas}}\right]+1$$

ori, dacă n este pozitiv, şi nu se execută dacă n este negativ. (Prin [ ] s-a notat valoarea întreagă a numărului.) Astfel, instrucţiunea:

for k = 5:4:83

se execută de:

$$n=\left[\frac{83-5}{4}\right]+1=\left[\frac{78}{4}\right]+1=19+1=20$$

Prin aceasta, valorile lui k sunt: 5, 9, 13, ..., 81. Bucla nu se va executa pentru valori

mai mari decât 83.

Exemplul 5.1.8. Să se calculeze funcţia:

$$f(x)=\begin{cases} 2x+3, & \text{dacă } x \in [-10, 2] \\ 2x^2-1, & \text{dacă } x \in (2, 20] \end{cases}$$

pentru toate valorile întregi pentru care este definită şi să se reprezinte grafic.
Secvenţa MATLAB corespunzătoare este:

```
x = -10:20
for k=1:length(x)
      if x(k)<=2
            f(k) = 2*x(k)+3
      else
            f(k) = 2*x(k)^2 -1
      end
end
plot(x,f)
```

Exemplul 5.1.9. Să se genereze o matrice Hilbert de ordinul n=4, ale cărei elemente sunt date de expresia:

$$H(i,j)=\frac{1}{i+j-1}$$

Cu secvenţa MATLAB: :

```
n=4
for i=1:n,
      for j=1:n,
            H(i,j)=1/(i+j-1);
      end
end
H
```

se obţine rezultatul :

```
H =
  1.0000   0.5000   0.3333   0.2500
  0.5000   0.3333   0.2500   0.2000
  0.3333   0.2500   0.2000   0.1667
  0.2500   0.2000   0.1667   0.1429
```

Exemplul 5.1.10. Să se calculeze şi să se reprezinte grafic funcţia:
f(x) = sin(x), pe intervalul [-2π, 2π], cu pasul 0.1.
Programul MATLAB este:

```
a=-2*pi; b=2*pi; c=0.1;
n=a:c:b;
k=max(size(n));
for i=1:k
      x(i)=sin(n(i));
end
plot(n,x); grid
```


5.1.3. Instrucţiunea repetitivă „while"

Instrucţiunea *while* este o structură care se utilizează pentru repetarea unui set de instrucţiuni, atâta timp cât o condiţie specificată este adevărată. Formatul general al acestei instrucţiuni de control este următorul:

while expresie
grup_de_instrucţiuni
end

Grupul de instrucţiuni se execută cât timp „expresie" are toate elementele nenule. „expresie" este de obicei sub forma:

expresie_1 condiţie expresie_2

unde „condiţie" este unul dintre operatorii relaţionali: ==, <, >, <=, >=, ~=.
Când condiţia este verificată (expresia este adevărată logic), se execută grupul de instrucţiuni. După ce se execută grupul de instrucţiuni, condiţia este retestată. Când condiţia este din nou adevărată, grupul de instrucţiuni se execută iar. Când condiţia este falsă, se trece la următoarea instrucţiune de după instrucţiunea *end*. Dacă expresia este totdeauna adevărată logic (valoarea acesteia este diferită de zero), bucla devine infinită; dintr-o buclă infinită se iese forţat prin apăsarea concomitentă a tastelor **[Ctrl]+[C]**.

Exemplul 5.1.11. Să se scrie un program, utilizând o buclă *while*, care calculează suma elementelor vectorului: x=[5 2 -9 10 -1 9 -1] până când întâlneşte un număr mai mare ca 8.
O variantă de program este următoarea:

```
x=[5 2 -9 10 -1 9 -1];
sum=0;
k=1;
while (x(k) <= 8)&(k <= length(x))
      sum = sum +x(k)
      k=k+1
end
```

Se obţine rezultatul:

```
sum=   -2
```

ieşirea din buclă realizându-se după ce au fost însumate primele trei elemente, al patrulea element neîndeplinind condiţia impusă.

5.1.4. Instrucţiunea break

Instrucţiunea *break* se utilizează pentru a ieşi dintr-o buclă înainte ca aceasta să se fi terminat. Se recomandă a fi utilizată dacă o condiţie de eroare este detectată în interiorul unei bucle. Instrucţiunea *break* încetează execuţia ciclurilor *for* şi *while*. În cazul unor cicluri imbricate, *break* comandă ieşirea din ciclul cel mai interior. Se apelează cu sintaxa:

break

5.1.5. Instrucţiunea return

Instrucţiunea *return* comandă o ieşire normală din fişierul-M către funcţia care l-a apelat sau către tastatură. Se apelează cu sintaxa:

return

5.1.6. Instrucţiunea error

Instrucţiunea *error* permite afişarea unor mesaje la întâlnirea unei erori. Se apelează cu sintaxa:

error('mesaj')

După afişarea textului „mesaj" controlul este redat claviaturii.

Exemplul 5.1.12. Procedura următoare verifică dacă funcţia test a fost apelată cu două argumente de intrare şi semnalează eroare dacă nu este îndeplinită această condiţie:

```
function test(x,y)
if nargin~=2
        error('Numărul argumentelor de intrare greşit.')
end
```

5.2. Funcţii de control logic

Funcţiile de control logic în MATLAB sunt:

exist	Verifică dacă variabilele sau funcţiile argument sunt definite;
any	Testează dacă cel puţin un element al unei matrice verifică o condiţie logică;
all	Testează dacă toate elementele unei matrice verifică o condiţie logică;
find	Returnează indicii elementelor diferite de zero;
isnan	Testează dacă elementele unei matrice sunt NaN;
isinf	Testează dacă elementele unei matrice sunt infinite;
finite	Testează dacă elementele unei matrice sunt finite.

MATLAB-ul conţine funcţii pentru testarea unor condiţii relaţionale şi logice în tablouri de date sau matrice. Acestea sunt des utilizate asociate cu instrucţiunea *„if"*.

5.2.1. Testarea condiţiilor logice

Funcţia *any* testează dacă un vector are cel puţin un element diferit de zero; returnează 1 dacă condiţia este verificată şi 0 în caz contrar. În cazul matricelor, funcţia *any* operează pe coloane, returnând un vector linie cu elemente 0 şi 1 care respectă condiţia de mai sus. Pentru fiecare coloană este returnat câte un element. Se apelează cu sintaxa:

Y=*any*(X)

Funcţia *all* testează dacă un vector are toate elementele diferite de zero; returnează 1 dacă condiţia este verificată şi 0 în caz contrar. În cazul matricelor, funcţia *all* operează pe coloane, returnând un vector linie cu elemente 0 şi 1 care respectă condiţia de mai sus. Pentru fiecare coloană este returnat câte un element. Se apelează cu sintaxa:

Y=*all*(X)

Funcţiile *any* şi *all* sunt utile mai ales în expresii de tip *if*. Spre exemplu, structura MATLAB:

```
if all(A < 0.5)
        instrucţiuni
end
```

permite realizarea calculelor prevăzute în „instrucţiuni" pentru toate elementele care îndeplinesc condiţia A < 0.5.

Exemplul 5.2.1. Fie $X = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 4 \end{bmatrix}$.

Să se determine:

- coloanele în care matricea are cel puţin un element diferit de zero;
- coloanele în care matricea are toate elementele diferite de zero;
- coloanele în care matricea are cel puţin un element mai mare ca -1.

Cu secvenţa MATLAB:

```
X=[1 0 2;0 0 1;0 0 4]
A=any(X)
B=all(X)
C=any(X>-1)
```

se obţin rezultatele:

```
A = [ 1   0   1 ]
B = [ 0   0   1 ]
C = [ 1   1   1 ]
```

5.2.2. Determinarea indicilor elementelor care verifică o condiţie

Funcţia *find* returnează indicii elementelor diferite de zero sau care verifică o condiţie impusă de utilizator. Se apelează cu una dintre sintaxele:

a=*find*(X) [l,c]=*find*(X) [l, c, V]=*find*(X)

Dacă argumentul X este un vector, prima formă returnează indicii elementelor lui X care sunt diferiţi de zero.

Dacă argumentul este o matrice, funcţia *find* analizează coloană cu coloană, indicii returnaţi ca rezultat având valorile cuprinse în intervalul 1 ... mxn (pentru o matrice cu m linii şi n coloane).

A doua formă de apelare returnează numărul liniei (vectorul l) şi al coloanei (vectorul c) matricei argument X care conţin elemente diferite de zero.

A treia formă de apelare returnează în plus şi vectorul V al elementelor diferite de zero (elimină elementele egale cu zero).

În unele situaţii este necesară determinarea poziţiei indicilor unor elemente dintr-o matrice care îndeplinesc o condiţie, de exemplu sunt mai mari sau mai mici decât o valoare

dată. Astfel structura MATLAB:

i = *find*(X >= a)

returnează indicii elementelor X mai mari sau egali cu a.

Exemplul 5.2.2. Fiind dată matricea $M = \begin{bmatrix} -7 & 0 & 0.4 \\ 3 & -5 & -2.5 \end{bmatrix}$,

să se găsească elementele a căror valoare este mai mică sau egală cu -2.5, precum şi indicii acestora.

Cu secvenţa MATLAB:

```
M = [ -7 0 0.4; 3 -5 -2.5]
i=find(M <= -2.5);
disp('    Indice      Valoarea')
a=[i   M(i)];
disp(a)
```

se obţine rezultatul:

Indice	Valoarea
1.0000	-7.0000
4.0000	-5.0000
6.0000	-2.5000

5.2.3. Determinarea poziţiei elementelor finite, infinite sau NaN

Determinarea poziţiei elementelor finite se face cu funcţia *finite*; se apelează cu sintaxa:

Y=*finite*(X)

care returnează o matrice Y de aceleaşi dimensiuni cu matricea argument X, cu elementele 1 pe poziţia în care elementele matricei X sunt numere finite şi zero în caz contrar.

Determinarea poziţiei elementelor NaN (Not-a-Number) se face cu funcţia *isnan;* se apelează cu sintaxa:

Y=*isnan*(X)

care returnează o matrice Y de aceleaşi dimensiuni cu matricea argument X, cu elementele 1 pe poziţia în care elementele matricei X sunt NaN şi zero în caz contrar. NaN (Not-a-Number) este rezultatul împărţirii 0/0.

Determinarea poziţiei elementelor infinite se face cu funcţia *isinf*; se apelează cu sintaxa:

Y=*isinf*(X)

care returnează o matrice Y de aceleaşi dimensiuni cu matricea argument X, cu elementele 1 pe poziţia în care elementele matricei X sunt Inf şi zero în caz contrar.

Exemplul 5.2.3. Să se scrie o secvenţă de program care înlocuieşte elementele NaN cu elemente zero în vectorul x=[0 2 NaN -89 Inf NaN].

Cu următoarea secvenţă:

```
x=[0 2 NaN -89 Inf NaN]
i = find(isnan(x));
x(i) = zeros(size(i))
```


se obţine rezultatul:

x = [0 2 0 -89 Inf 0]

Exemplul 5.2.4. Aplicaţi funcţiile *isnan, isinf* şi *finite*.

Cu secvenţa MATLAB:

```
X = [ 0/0     0/1  2 1/0 ]
Y=[ ]
ax = isnan(X)
bx = finite(X)
cx = isinf(X)
```

rezultă:

X = [NaN 0 2 ∞]
ax = [1 0 0 0]
bx = [0 1 1 0]
cx = [0 0 0 1]

5.2.4. Testarea matricelor goale

Testarea unei matrice dacă este goală se face cu funcţia *isempty;* se apelează cu sintaxa:

v=*isempty*(X)

care returnează 1 dacă matricea X este goală şi zero în caz contrar.

5.2.5. Testarea existenţei unei variabile

Testarea existenţei unei variabile sau a unui fişier se face cu funcţia *exist*; se apelează cu sintaxa:

v=*exist*('nume_variabilă')

care returnează următoarele valori:

0 - dacă nu există ;
1 - dacă există ca o variabilă în spaţiul de lucru;
2 - dacă există ca un fişier-M pe disc;
3 - dacă există ca un fişier-MEX pe disc;
4 - dacă există ca funcţie SIMULINK;
5 - dacă există ca funcţie MATLAB internă.

Numele variabilei trebuie să fie scris între apostrofuri, fiind şir de caractere. Funcţia *exist* se poate introduce într-un program pentru a testa dacă anumite fişiere există pe disc, atunci când se lansează un program. De exemplu secvenţa:

```
h=exist('test')
if h==0
      error('Lipseste fisierul test.m')
end
```

verifică esistenţa fişierului test.m şi returnează un mesaj de avertizare dacă acesta nu există.

5.3. Vectorizarea calculelor

Deoarece operaţiile cu vectori şi matrice sunt executate în MATLAB mai repede cu un ordin de mărime decât operaţiile compilate/interpretate, se obţine o viteză de lucru mai mare dacă algoritmii încrişi în fişierele-M sunt vectorizaţi. Oriunde este posibil, ciclurile *for* şi *while* trebuie convertite în operaţii cu vectori sau matrice.

Spre exemplu, programul care calculează sinusul în 1000 de puncte (de la 1 la 10, cu pasul .01), utilizând bucla *for*:

```
t=0:.01:10;
N=length(t)
for i=1:N;
      y(i)=sin(t(i));
end
plot(t,y)
```

are versiunea vectorizată:

```
t=0:.01:10;
Y=sin(t);
plot(t,Y)
```

Primul exemplu necesită un timp 10-15 ori mai mare decât cel de-al doilea.

În cazurile în care nu se poate vectoriza o parte din program, pentru a face ca ciclurile să fie executate mai repede, se procedează la prealocarea unor vectori în care vor fi reţinute rezultatele. De exemplu, includând o primă instrucţiune de prealocare folosind funcţia *zeros*, ciclul *for* următor se execută semnificativ mai repede:

```
y=zeros(1,100);
for i=1:100
      y(i)=det(x^i);
end
```

Explicaţia constă în faptul că dacă nu se face prealocarea, interpretorul MATLAB trebuie să redimensioneze vectorul y la o dimensiune mai mare, de fiecare dată când trece printr-o iteraţie a ciclului. Dacă vectorul este prealocat, acest pas este eliminat şi execuţia este mai rapidă.

Pentru lucrul cu matrice mari pe computere cu memorie limitată, prealocarea are un al doilea avantaj: utilizarea memoriei mult mai eficient şi lansarea programelor fără a depăşi memoria. Prealocarea ajută inclusiv la reducerea fragmentării memoriei; o memorie fragmentată în cursul sesiunii MATLAB poate deveni insuficientă ca spaţiu continuu necesar memorării variabilelor mari.

6. OPERAREA CU ŞIRURI DE CARACTERE

Constanta şir se reprezintă printr-un şir de octeţi în care se păstrează codurile ASCII ale caracterelor şirului respectiv. Codurile ASCII 0-31 sunt caractere negrafice, iar codurile 32-127 sunt caractere grafice. Un şir de caractere este constituit din unul sau mai multe caractere delimitate prin apostrofuri. Apostrofurile nu fac parte din şirul de caractere.

În acest capitol se prezintă declararea şirurilor de caractere, compararea acestora, conversia şirurilor în numere şi a numerelor în şiruri, evaluarea expresiilor MATLAB din şiruri concatenate etc.

6.1. Funcţii generale

Funcţiile MATLAB folosite pentru operarea cu şiruri de caractere sunt:

abs	Converteşte şirurile de caractere în valorile numerice ale codului ASCII;
setstr	Converteşte valorile numerice ale codului ASCII în caractere;
strings	Declară o variabilă ca şir de caractere;
isstr	Detectează dacă o variabilă este şir de caractere;
str2mat	Formează matrice text din şiruri de caractere;
blanks	Creează spaţii între şirurile de caractere;
deblank	Elimină spaţiile libere dintre şirurile de caractere;
eval	Evaluează şirurile de caractere cu expresii MATLAB;
feval	Evaluează funcţiile MATLAB.

Funcţia *abs* converteşte şirurile de caractere la valorile numerice ale codului ASCII; se apelează cu sintaxa:

y=abs('şir_de_caractere')

Exemplul 6.1.1. Să se determine codurile ASCII ale şirului de caractere „MATLAB".
Cu secvenţa :

```
y=abs('MATLAB')
```

se obţine : y = [77 65 84 76 65 66]

Funcţia *setstr* setează indicatorul şir de caractere; se apelează cu sintaxa:

s=*setstr*(t)

Rezultatul s este un şir de caractere ASCII care corespund valorilor numerice t. Caracterele ASCII care se pot tipări sunt întregi în intervalul 32...127 sau în intervalul 0...255 dacă se folosesc 8 biţi. În afara acestui interval, numerele sunt rotunjite la întregi modulo 256.

Exemplu 6.1.2. Să se afişeze caracterele ASCII grafice (codurile ASCII 32-127), într-o matrice 3x32.
Cu secvenţa MATLAB:

```
ascii=setstr(reshape(32:127,32,3)')
```

se obţine rezultatul :

```
ascii =
    !"#$%&'()*+,-./0123456789:;<=>?
    @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
    `abcdefghijklmnopqrstuvwxyz{|}~□
```

Exemplul 6.1.3. Să se scrie o buclă infinită care determină un apel sonor (foloseşte caracterul ASCII cu codul 7).
Cu secvenţa :

```
while 1
disp(setstr(7))
pause(2)
end
```

se obţine efectul dorit.

Funcţia *strings* returnează un identificator al şirurilor de caractere.
Instrucţiunea :

t = 'MATLAB'

creează un vector ale cărui componente sunt codurile ASCII ale şirului caracterelor MATLAB. Pentru a afişa codul numeric ASCII se foloseşte instrucţiunea:

n=*abs*(t)

iar pentru a reveni în formatul text, se foloseşte instrucţiunea :

t=*setstr*(n)

Detectarea şirurilor de caractere se face cu funcţia *isstr*, care se apelează cu sintaxa :

a= *isstr*(t)

Returnează 1 pentru modul text şi 0 pentru modul cod numeric ASCII.

Introducerea unui apostrof într-un text se face folosind două apostrofuri succesive. Un şir compus din subşiruri este realizat prin concatenarea acestora, după structura :

şir = ['subşir1', 'subşir2', ... , 'subşir n']

Funcţia *str2mat* formează matrice şir de caractere din şiruri individuale; se apelează cu sintaxa:

S=*str2mat*(s1, s2, ...)

Numărul de linii al matricei S este egal cu numărul de şiruri individuale, iar numărul de coloane este egal cu dimensiunea celui mai mare şir component. Prin acest procedeu se pot construi matrice cu până la 11 şiruri. De asemenea, fiecare argument de intrare, poate fi la rândul lui o matrice şir, fiind posibilă operarea cu matrice şir de dimensiuni mari.

Exemplul 6.1.4. Creaţi o matrice şir cu subşirurile: 'Pachetul de programe', 'MATLAB', 'este performant.'.
Cu secvenţa:

```
S=str2mat('Pachetul de programe',...
'''MATLAB''','este performant.')
```

se obţine rezultatul:


```
S=
    Pachetul de programe
    'MATLAB'
    este performant.
```

Funcţia *blanks* se foloseşte pentru afişarea şirurilor la anumite poziţii sau pentru crearea unor spaţii între şiruri; se apelează cu sintaxa:

blanks(n)

care deplasează cursorul cu n caractere spre dreapta (n număr întreg pozitiv). De cele mai multe ori se foloseşte asociată cu funcţia *disp* şi permite controlul liniei şi coloanei în care se face afişarea unui şir.

Exemplul 6.1.5. Să se scrie şirurile din exemplul anterior formatate astfel: al doilea deplasat la dreapta cu 10 caractere faţă de primul, iar al treilea cu o linie liberă faţă de al doilea şi cu 20 caractere deplasare spre dreapta faţă de primul.
Cu secvenţa:

```
disp('Pachetul de programe')
disp([blanks(10),''MATLAB''])
disp(blanks(1)')
disp([blanks(20),'este performant.'])
```

se obţine rezultatul:

```
Pachetul de programe
        'MATLAB'

                    este performant.
```

Funcţia *deblank* elimină spaţiile libere şi orice caracter nul (cele cu valoarea absolută 0) de la sfârşitul unui şir de caractere; se apelează cu sintaxa:

deblank(S)

Funcţia *eval* interpretează şirurile de caractere conţinând expresii MATLAB; se apelează cu una dintre sintaxele :

x=*eval*('şir_de_caractere') *eval*('şir_de_caractere')

Dacă 'şir_de_caractere'este o expresie MATLAB, aceasta este interpretată. De cele mai multe ori 'şir_de_caractere' concatenează mai multe subşiruri şi variabile cuprinse între paranteze drepte, construindu-se expresii MATLAB în programe complexe.

Exemplul 6.1.6. Să se evalueze expresia: $y=2x^2+3$, folosind funcţia *eval*.
Cu secvenţa MATLAB:

```
x=3; z='2*x^2+3';
y=eval(z)
```

se obţine rezultatul : y = 21

Exemplul 6.1.7. Să se genereze 3 matrice cu elemente numere aleatoare cu numele test1, test2, test3.
Secvenţa corespunzătoare este:

```
for n = 1:3
    eval(['test' int2str(n) '=rand(4,4)'])
    pause
```

```
end
```

Exemplul 6.1.8. Să se scrie o secvenţă de program care încarcă şi prelucrează datele din toate fişirele test1.dat, test2.dat, ... existente pe disc.
Cu secvenţa :

```
k=0
while 1
    k=k+1
    testk=['test' int2str(k)];
    nume=[testk '.dat'];
    if ~exist(nume), break, end
       eval(['load ' nume]);
       X=eval(testk);
      %Secventa de prelucrare date
end
```

se realizează scopul propus.

Funcţia *feval* permite evaluarea funcţiilor; se apelează cu una dintre sintaxele :

feval('nume_funcţie', x1,..., xn)

[y1,...ym]=*feval*('nume_funcţie', x1, ..., xn)

unde:

nume_funcţie - este un argument şir, care specifică numele funcţiei de evaluat;
x1,... xn - sunt argumentele de intrare;
y1,... ym - sunt argumentele de ieşire.

Spre exemplu, funcţia test.m cu două argumente de intrare şi un argument de ieşire, se apelează cu sintaxa:

y=*feval*('test', x1, x2)

6.2. Compararea şirurilor de caractere

Funcţiile MATLAB folosite pentru compararea şirurilor de caractere sunt:

strcmp Compară şiruri de caractere;
findstr Caută un şir de caractere într-un alt şir de caractere;
upper Converteşte literele unui şir de caractere la litere mari;
lower Converteşte literele unui şir de caractere la litere mici;
isletter Verifică dacă elementele unui vector sunt sau nu literele alfabetului.

Compararea a două şiruri de caractere se face cu funcţia *strcmp*, care se apelează cu sintaxa :

a=*strcmp*(S1,S2)

Returnează 1 dacă acestea sunt identice şi 0 în caz contrar. Funcţia lucrează în modul senzitiv, incluzând în procesul de comparare spaţiile dintre caractere şi tipul caracterelor - mari sau mici - folosite la scriere.

Exemplul 6.2.1. Comparaţi şirurile „test" şi „Test" şi afişaţi rezultatul.
Cu secvenţa MATLAB:

```
a=strcmp('test','Test')
```


se obţine rezultatul : a= 0

Căutarea unui şir de caractere, S2, într-un alt şir de caractere, S1, se face cu funcţia *findstr*, care se apelează cu sintaxa:

M=*findstr*(S1, S2)

Rezultatul M conţine poziţiile de la care începe şirul S2 în şirul S1.

Exemplul 6.2.2. Să se găsească poziţia şirului de caractere S2='test' în şirul de caractere S1='Acest test este interesant'.
Cu secvenţa:

```
S1='Acest test este interesant';
S2='test';
M=findstr(S1,S2)
```

se obţine răspunsul: `M = 7`

Conversia literelor mici dintr-un şir de caractere, în litere mari, se face cu funcţia *upper*, care se apelează cu sintaxa:

t=*upper*(s)

iar conversia inversă, a literelor mari în litere mici se face cu funcţia *lower*, care se apelează cu sintaxa:

t=*lower*(s)

Determinarea literelor alfabetului dintr-un şir de caractere se face cu funcţia *isletter*, care se apelează cu sintaxa:

A=*isletter*(S)

Rezultatul conţine 1 în poziţiile în care elementele şirului sunt litere şi 0 în celelalte poziţii (numere, blank-uri, etc.).

Exemplul 6.2.3. Determinaţi poziţiile în care se găsesc literele din şirul 'Azi este 15.09.1995'.
Cu secvenţa:

```
S='Azi este 15.09.1995';
A=isletter(S)
B=find(A==1)
```

se obţine rezultatul:

A = [1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
B = [1 2 3 5 6 7 8]

6.3. Conversia şir-număr şi număr-şir

Funcţiile MATLAB folosite pentru conversia numerelor în şiruri de caractere sunt:

num2str Converteşte un număr într-un şir de caractere;
int2str Converteşte un număr întreg într-un şir de caractere;
str2num Converteşte un şir de caractere într-un număr;
sprintf Converteşte un număr într-un şir sub controlul formatului.

Conversia numerelor în şiruri de caractere este utilizată la introducerea valorilor numerice în titluri, la etichetarea axelor etc. Se face cu funcţia *num2str* care se apelează cu sintaxa:

s=*num2str*(x)

Funcţia *int2str* converteşte numerele întregi în şiruri de caractere; se apelează cu sintaxa :

s=*int2str*(n)

Funcţia *str2num* converteşte şirurile care au caracterele ASCII ale valorilor numerice, în numere; se apelează cu sintaxa :

x=*str2num*(s)

Şirurile pot conţine: numerele zecimale 0...9, semnale + şi -, punctul zecimal, caracterul e urmat de un număr (puterea lui 10), precum şi caracterele i şi j pentru numerele complexe.

Funcţia *sprintf* scrie datele din matricea (sau vectorul) A sub controlul formatului; se apelează cu sintaxa:

s=*sprintf*('format', A, ...)

Parametrul de control, şirul „format", conţine specificaţia de conversie şi poate fi de forma :

```
%-d.p tip_format
```

unde :

- semnul minus este opţional şi specifică alinierea la stânga (dacă lipseşte, alinierea se face la dreapta);
- d este numărul minim de câmpuri (cifre) afectate datei respective;
- p este numărul de câmpuri folosite pentru partea zecimală;
- tip_format poate fi unul dintre caracterele :
  - e - notaţia exponenţială;
  - f - notaţia în virgulă fixă;
  - g - formatele scurte ale lui e sau f (fără zerourile nesemnificative).

Caracterele de control admise de MATLAB sunt :

\n - linie nouă; \f - form feed;
\t - tab orizontal; \\ - backslash;
\b - backspace; \' - aprostrof.
\r - carriage return (la cap de rând);

Exemplul 6.3.1. Conversia numărului π în şir de caractere se face cu secvenţa:

```
s=num2str(pi)
```

obţinându-se rezultatul : s=3.142

Exemplul 6.3.2. Să se introducă în titlul unui grafic o valoare numerică rezultată dintr-un calcul.

Cu secvenţa :

```
a=2; b=8; s=a+b;
title(['Viteza este ', int2str(s),' m/s'])
```

se obţine titlul graficului: „Viteza este 10 m/s"

Exemplul 6.3.3. Conversia şirului de caractere s=123.25e-2 în număr se realizează cu următoarea secvenţă:

```
x=str2num('123.25e-2')
```

Exemplul 6.3.4. Să se tipărească elementele vectorului :
x = [1.12346578 5.123] în formatele exponenţial şi cu virgulă fixă, folosind 6 digiţi, dintre care 5 pentru partea zecimală.

Cu secvenţa MATLAB:

```
x=[1.12345678 5.123];
y1=sprintf('%6.5e\t %6.5e',x)
y2=sprintf('%6.5g\t %6.5g',x)
y3=sprintf('%6.5f\n %6.5f',x)
```

se obţin rezultatele :

```
y1 = [1.12346e+000      5.12300e+000 ]
y2 = [ 1.1235      5.123 ]
y3 =
        1.12346
        5.12300
```

6.4. Conversia şirurilor în formate numerice

Funcţiile MATLAB folosite pentru conversia şirurilor în formate numerice sunt:

dec2hex Converteşte un întreg zecimal într-un şir hexa;
hex2dec Converteşte un şir hexa într-un întreg zecimal.

Funcţia *dec2hex* converteşte numerele zecimale întregi la numere hexazecimale, memorate în şiruri MATLAB; se apelează cu sintaxa :

s=*dec2hex*(n)

Funcţia *hex2dec* converteşte numerele hexazecimale în numere întregi, memorate în şiruri MATLAB, ca numere zecimale; se apelează cu sintaxa :

d=*hex2dec*('s ')

Exemplul 6.4.1. Să se convertească în numere hexazecimale numerele zecimale: 15 şi 256.

Cu secvenţa MATLAB:

```
x=dec2hex(15)
y=dec2hex(256)
```

se obţine rezultatul : x= F y= 100

Exemplul 6.4.2. Să se convertească în numere zecimale numerele hexazecimale F şi 100.

Cu secvenţa MATLAB:

```
a=hex2dec('F')
b= hex2dec('100')
```

se obţine rezultatul : a= 15 b= 256

Partea a II-a
CALCUL NUMERIC CU MATLAB

7. OPERAŢII ARITMETICE

Calculele aritmetice asupra tablourilor de date în MATLAB pot fi:

- operaţii după regulile calculului matriceal - operaţii cu matrice;
- operaţii după regulile calculului scalar - operaţii cu tablouri.

Operatorii folosiţi în calculele aritmetice cu tablouri şi matrice sunt prezentaţi în tabelul 7.1:

Tabelul 7.1. Operatorii aritmetici MATLAB

Operaţia	Scalari	Matrice	Tablouri
Adunarea	+	+	+
Scăderea	-	-	-
Înmulţirea	*	*	.*
Împărţirea la stânga	\	\	.\
Împărţirea la dreapta	/	/	./
Ridicarea la putere	^	^	.^
Transpunerea	'	'	.'

7.1. Operaţiile aritmetice cu scalari

Operaţiile aritmetice între doi scalari sunt prezentate în tabelul 7.2.

Tabelul 7.2. Forma MATLAB a operaţiilor cu scalari

Operaţia	Forma algebrică	Forma MATLAB
Adunare	a+b	a+b
Scădere	a-b	a-b
Înmulţire	axb	a*b
Împărţire la dreapta	a:b	a/b
Împărţire la stânga	b:a	a\b
Ridicare la putere	a^b	a^b

Expresiile aritmetice pot fi evaluate şi rezultatul memorat în variabile specificate. Astfel, instrucţiunea:

x=a+b

atribuie variabilei x, suma dintre variabilele a şi b.
Instrucţiunea:

k=k+1

atribuie variabilei k o nouă valoare, egală cu suma dintre vechea valoare şi constanta 1.
În urma instrucţiunilor succesive:

```
a=1
a=2.5
```

în variabila a se află valoarea 2.5.

O valoare introdusă fără nominalizare este asignată variabilei ans (answer). În variabila ans este memorată în permanenţă valoarea ultimei variabile căreia nu i s-a atribuit un nume.

7.1.1. Ordinea operaţiilor aritmetice

Ordinea operaţiilor în MATLAB, tabelul 7.3, este aceeaşi cu cea a operaţiilor aritmetice standard, cunoscută în matematica elementară.

Tabelul 7. 3. Ordinea operaţiilor aritmetice

Ordinea	Operaţia
1	parantezele
2	ridicarea la putere
3	înmulţirea şi împărţirea
4	adunarea şi scăderea

Exemplul 7.1.1. Să se evalueze funcţia :

$$f(x) = \frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.5x - 1}$$

pentru x=2.

O variantă de program MATLAB, este :

```
x=2
M=x^3-2*x^2+x-6.3
N=x^2+0.5*x-1
f=M/N
```

dar este posibilă şi o formă mai compactă:

```
x=2
f=(x^3-2*x^2+x-6.3)/(x^2+0.5*x-1)
```

7.1.2. Limitele calculelor

Deşi variabilele memorate au un interval foarte mare, cel mai adesea fiind între limitele 10^{-308} şi 10^{308} (vezi funcţiile MATLAB *realmax* şi *realmin*), totuşi, uneori este posibil ca rezultatul unei expresii să depăşească aceste limite, aşa cum se întâmplă în exemplul:

$$x = 2.5 \cdot 10^{200}$$

$$y = 10^{200}$$

$$z = x \cdot y = 2.5 \cdot 10^{400}$$

Deoarece z este în afara limitelor de mai sus, valoarea calculată z nu poate fi memorată. MATLAB-ul înregistrează ∞. Se verifică acest lucru cu secvenţa:

```
x=2.5*10^200
y=10^200
```



```
z=x*y
```

care returnează:

```
z =  Inf
```

Rezultatul unui calcul este mai mic decât 10^{-308} calculatorul înregistrează valoarea zero.

În MATLAB rezultatul împărţirii cu zero este ∞. În acest caz se afişează mesajul de atenţionare „Warning:Divide by zero", dar calculele continuă cu operandul ∞.

7. 2. Operaţiile aritmetice cu tablouri

Operaţiile cu tablouri sunt operaţii aritmetice (înmulţire, împărţire, ridicare la putere, etc) între elementele situate în aceeaşi poziţie a tablourilor, cunoscute sub numele de operaţii element cu element.

Pentru a preciza că înmulţirea se efectuează element cu element între componentele a două matrice de aceleaşi dimensiuni, se utilizează operatorul de înmulţire precedat de punct (.*), adică:

```
C=A.*B
```

Pentru efectuarea operaţiilor cu tablouri se folosesc aceiaşi operatori ca în operaţiile cu scalari, precedaţi de semnul punct „ .", semn ce indică efectuarea operaţiilor în ordinea element cu element. Dacă unul dintre operanzi este un scalar, acesta operează cu fiecare element al tabloului.

7.2.1. Adunarea şi scăderea

Exemplul 7.2.1. Fie: A=[2 5 6], B=[4 3 2], p=2.
Să se calculeze: C=A-B, D=A-p şi E=p-A
Cu secvenţa MATLAB:

```
A=[2 5 6]; B=[4 3 2]; p=2;
C=A-B
D=A-p
E=p-A
```

se obţin rezultatele:

```
C = [-2  2  4]
D = [ 0  3  4]
E = [ 0 -3 -4]
```

7.2.2. Înmulţirea tablourilor

Exemplul 7.2.2. Fie: A=[2 5 6], B=[4 3 2], p=2.
Să se calculeze : C=A.*B, D=A.*p şi E=p.*A .
Cu secvenţa MATLAB:

```
A=[2 5 6]; B=[4 3 2];p=2;
C=A.*B
D=A.*p
E=p.*A
```

se obţin rezultatele:

C = [8 15 12]
D = [4 10 12]
E = [4 10 12]

7.2.3. Împărţirea la dreapta

Operaţia de împărţire la dreapta, element cu element, între două tablouri este simbolizată cu operatorul punct-slash (./). Instrucţiunea:

Z = X./Y

reprezintă împărţirea element cu element a tablourilor X şi Y, rezultând elementele:

Z(i,j) = X(i,j)/Y(i,j)

Exemplul 7.2.3. Fie: A=[2 5 6], B=[4 3 2], p=2.
Să se calculeze : C=A./B, D=A./p şi E=p./A.
Cu secvenţa MATLAB:

```
A=[2 5 6]; B=[4 3 2]; p=2;
C=A./B
D=A./p
E=p./A
```

se obţine rezultatul:

C = [0.5000 1.6667 3.0000]
D = [1.0000 2.5000 3.0000]
E = [1.0000 0.4000 0.3333]

7.2.4. Împărţirea la stânga

Operaţia de împărţire la stânga, element cu element, între două tablouri este simbolizată cu operatorul punct-bakslash (.\ ). Instrucţiunea:

Z = X.\Y

reprezintă împărţirea element cu element a tablourilor X şi Y, cu aceleaşi dimensiuni, rezultând un tablou cu elementele:

Z(i,j) = X(ij)\Y(ij) = Y(i,j)/X(i,j)

Prin urmare:

Z = X.\Y = Y./X

Exemplul 7.2.4. Fie: A=[2 5 6] B=[4 3 2] p=2.
Să se calculeze : C=A.\B , D=A.\p şi E=p.\A
Cu secvenţa MATLAB:

```
A=[2 5 6]; B=[4 3 2]; p=2;
C=A.\B
D=A.\p
E=p.\A
```

se obţin rezultatele:

C = [2.0000 0.6000 0.3333]
D = [1.0000 0.4000 0.3333]
E = [1.0000 2.5000 3.0000]

7.2.5. Ridicarea la putere

Operaţia de ridicare la putere element cu element într-un tablou este simbolizată cu operatorul punct-^ (.^). Următoarea instrucţiune:

Z=X.^Y

reprezintă ridicarea fiecărui element din tabloul X la puterea indicată de valoarea elementului din aceeaşi poziţie a tabloului Y, adică:

Z(i,j) = X(i,j)^Y(i,j)

Dacă X este un scalar, se lasă un blanc între scalar şi operatorul de ridicare la putere „ .^ " pentru a nu interpreta punctul care indică operarea cu tablouri de elemente ca punctul zecimal.

Exemplul 7.2.5. Fie: A=[2 5 6], B=[4 3 2], p=2.
Să se calculeze : C=A.^B, D=A.^p şi E=p.^A.
Cu secvenţa MATLAB:

```
A=[2 5 6]; B=[4 3 2]; p=2;
C=A.^B
D=A.^p
E=p.^A
```

se obţin rezultatele:

C = [16 125 36]
D = [4 25 36]
E = [4 32 64]

7.2.6. Transpunerea tablourilor

Operaţia de transpunere a unui tablou este simbolizată de operatorul punct-apostrof. Cu instrucţiunea:

Z=Y.'

liniile tabloului Y devin coloanele tabloului transpus Z. Acest lucru face ca un tablou Y cu dimensiunea m x n să devină un tablou Z cu dimensiunea n x m.

Exemplul 7.2.6. Să se determine transpusa tabloului:

$$Z = \begin{bmatrix} 1 & 1+i \\ i & 3-2*i \end{bmatrix}$$

Cu secvenţa MATLAB:

```
Z=[1 1+i; i 3-2*i];
Z1=Z.'
```

se obţin rezultatele:

Z1 =
1.0000 0 + 1.0000i
1.0000 + 1.0000i 3.0000 - 2.0000i

7.3. Operaţiile aritmetice cu vectori

7.3.1. Produsul scalar

Produsul scalar a doi vectori de aceeaşi dimensiune este un scalar, egal cu suma produselor corespunzătoare aceloraşi poziţii:

$$C = \vec{A} \cdot \vec{B} = \sum_{i=1}^{N} a(i) \cdot b(i)$$

Produsul scalar a doi vectori, $\vec{A}$ şi $\vec{B}$, se calculează cu instrucţiunea:

C=*sum*(A.*B)

Dacă A este de dimensiunea 1xN şi B de dimensiunea Nx1, atunci sunt posibile următoarele expresii pentru calculul produsului scalar:

E=sum(A'.*B) F=sum(A.*B')

Cosinusul unghiului vectorilor:

$$\vec{A} = a_x \vec{i} + a_y \vec{j} + a_z \vec{k}$$

$$\vec{B} = b_x \vec{i} + b_y \vec{j} + b_z \vec{k}$$

se calculează cu relaţia:

$$\cos(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{|A| \cdot |B|} = \frac{a_x b_x + a_y b_y + a_z c_z}{\sqrt{a_x^2 + a_y^2 + a_z^2}\sqrt{b_x^2 + b_y^2 + b_z^2}}$$

unde $|A|$ şi $|B|$ sunt lungimile, norma sau modulul. Dacă produsul scalar este nul, cei doi vectori sunt ortogonali.

Exemplul 7.3.1. Să se calculeze produsul scalar şi unghiul dintre vectorii:

$$\vec{a} = 3\vec{i} - 4\vec{j} \quad \text{şi} \quad \vec{b} = \vec{i} + 2\vec{j} - 2\vec{k}$$

Cu secvenţa MATLAB:

```
a=[3  -4  0];  b=[1  2  -2];
ab=sum(a.*b);  %produsul scalar
mod_a=norm(a)
mod_b=norm(b)
alfa=acos(ab/(mod_a*mod_b))*180/pi
```

se obţine rezultatul:

ab= -5

alfa = 109.4712 grade

7.3.2. Produsul vectorial

Produsul vectorial a doi vectori $\vec{A}$ şi $\vec{B}$:

$$\vec{C} = \vec{A} \times \vec{B} = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix}$$

este un vector perpendicular pe planul celor doi vectori. Modulul produsului vectorial este:

$$|C| = |A| \cdot |B| \cdot \sin(\vec{A}, \vec{B})$$

De exemplu, dacă A şi B descriu vectorii cu ajutorul coordonatelor (proiecţiile după axele Ox, Oy şi Oz), cu secvenţa MATLAB:

```
A=[ax ay az];   B=[bx by bz];
C= A'*B
```

se obţine ca rezultat matricea C, cu dimensiunea 3x3.

$$\vec{C} = \begin{array}{c} A \backslash B \\ \vec{i} \\ \vec{j} \\ \vec{k} \end{array} \begin{array}{c} \begin{array}{ccc} \vec{i} & \vec{j} & \vec{k} \end{array} \\ \begin{bmatrix} x & \vec{k} & -\vec{j} \\ -\vec{k} & x & \vec{i} \\ \vec{j} & -\vec{i} & x \end{bmatrix} \end{array}$$

Componentele C(i,j) au următoarea semnificaţie:

- pentru i≠j reprezintă un vector orientat după versorul menţionat la poziţia respectivă a matricei C;
- pentru i=j reprezintă o componentă a produsului scalar (marcată cu x în matricea C)

Vectorul rezultant $\vec{C}$ are componentele:

```
cx=C(2,3)-C(3,2)
cy=C(3,1)-C(1,3)
cz=C(1,2)-C(2,1)
PV=[cx cy cz];  %Produsul vectorial
```

Suma elementelor diagonalei principale constituie produsul scalar al vectorilor $\vec{A}$ şi $\vec{B}$, şi se calculează cu instrucţiunea MATLAB:

```
PS=sum(diag(C)); %Produsul scalar
```

Prin urmare, produsul a doi vectori este:

- un scalar, dacă operaţia este un produs scalar;
- o matrice, dacă operaţia este un produs vectorial

Exemplul 7.3.2. Să se calculeze produsul vectorial şi scalar al vectorilor:

$$\vec{a} = 5\vec{i} - 3\vec{j} - \vec{k} \quad \text{şi} \quad \vec{b} = -\vec{i} - 2\vec{j} - 2\vec{k}$$

Cu secvenţa MATLAB:

```
a=[5 -3 -1]; b=[-1 -1 -2];
C=a'*b;
cx=C(2,3)-C(3,2);
cy=C(3,1)-C(1,3);
cz=C(1,2)-C(2,1);
PV=[cx cy cz]   %produsul vectorial
PS1=sum(a.*b)   % produsul scalar metoda 1
PS2=sum(diag(C))%produsul scalar metoda 2
```

se obţin rezultatele:

```
PV = [5   11   -8]
PS1 =  0
PS2 =  0
```

care reprezintă vectorul produs vectorial: $\vec{a} = 5\vec{i} + 11\vec{j} - 8\vec{k}$; produsul scalar nul evidenţiază faptul că cei doi vectori sunt perpendiculari.

7.4. Operaţiile aritmetice cu matrice

Operaţiile uzuale de algebră liniară cu matrice sunt simbolizate cu semnele grafice: *, \, /, ^, ', şi se efectuează după regulile cunoscute din calculul matriceal.

7.4.1. Adunarea şi scăderea

Operaţia de adunare a două matrice este simbolizată cu operatorul plus (+). Instrucţiunea:

Z = X+Y

reprezintă adunarea matricelor X şi Y, rezultând elementele:

Z(i,j) = X(i,j)+Y(i,j).

Matricele X şi Y trebuie să aibă aceleaşi dimensiuni, în afara cazului când X sau Y este un scalar. Un scalar a poate fi adunat cu orice matrice, rezultând elementele:

Z(i,j) = a+X(i,j)

Operaţia de scădere a două matrice este simbolizată cu operatorul minus (-). Instrucţiunea:

Z = X-Y

reprezintă scăderea matricei Y din X, rezultând elementele:

Z(i,j) = X(i,j)-Y(i,j)

Matricele X şi Y trebuie să aibă aceleaşi dimensiuni, în afara cazului când X sau Y este un scalar. Un scalar a poate opera cu orice matrice, rezultând elementele:

Z(i,j) = X(i,j)-a

sau:

Z(i,j) = a-X(i,j)

Exemplul 7.4.1. Fie: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, $p = 2$.

Să se calculeze: A+B, A-B, A+p.

Cu secvenţa MATLAB:

```
A=[1 2;3 4]; B=[5 6;7 8]; p=2;
C1=A+B
C2=A-B
C3=A+p
```

se obţin rezultatele:

```
C1 =                C2 =                C3 =
   6   8               4  -4                3   4
  10  12               4  -4                5   6
```

7.4.2. Înmulţirea matricelor

Operaţia de înmulţire a două matrice este simbolizată cu operatorul steluţă (*). Următoarea instrucţiune:

Z = X*Y

reprezintă matricea produs a având elementele:

$$Z(i,j) = \sum_k X(i,k)Y(k,j)$$

Produsul matriceal este posibil dacă numărul coloanelor matricei X este egal cu numărul liniilor matricei Y, elementul Z(i,j) fiind suma produselor dintre elementele liniei i cu elementele corespondente din coloana j. Produsul matrice-vector este un caz special al cazului general al produsului matrice-matrice. De asemenea, un scalar poate poate fi înmulţit cu orice matrice, realizându-se înmulţirea cu fiecare element al matricei:

Z(i,j) = a*X(i,j).

Exemplul 7.4.2. Fie: $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, $B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, $p = 2$.

Să se calculeze: A*B, A*p.

Cu secvenţa MATLAB:

```
A=[1 2;3 4]; B=[5 6;7 8]; p=2;
C1=A*B
C2=A*p
```

se obţin rezultatele:

```
C1 =                C2 =
  19  22               2   4
  43  50               6   8
```

7.4.3. Împărţirea la dreapta

Operaţia de împărţire la dreapta a două matrice este simbolizată cu operatorul slash (/). Următoarea instrucţiune:

Z = X/Y

reprezintă împărţirea la dreapta a matricelor X şi Y, şi este identică cu $X*Y^{-1}$ (Y^{-1} este inversa matricei Y).

Exemplul 7.4.3. Fie: $A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$, $B=\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix}$.

Să se calculeze A/B.
Cu secvenţa MATLAB:

```
A=[1 2;3 4]; B=[5 6;7 8];
C=A/B
```

se obţine rezultatul:

```
C =
   3.0000  -2.0000
   2.0000  -1.0000
```

7.4.4. Împărţirea la stânga

Operaţia de împărţire la stânga a două matrice este simbolizată cu operatorul backslash (\ ). Următoarea instrucţiune:

Z = X\Y

reprezintă împărţirea la stânga a matricelor X şi Y şi este identică cu $X^{-1}*Y$ (Y^{-1} este inversa matricei Y).

Dacă A este o matrice n x n, iar B este un vector coloană cu n componente, atunci X=A\B este soluţia sistemului de ecuaţii AX=B, obţinută prin eliminarea Gauss.

Dacă unul dintre operanzi este scalar, operaţia nu este posibilă.

Exemplul 7.4.4. Fie : $A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$, $B=\begin{bmatrix}5 & 6\\ 7 & 8\end{bmatrix}$.

Să se calculeze A\B.
Cu secvenţa MATLAB:

```
A=[1 2;3 4];B=[5 6;7 8];
C=A\B
```

se obţine rezultatul:

```
C =
  -3  -4
   4   5
```

7.4.5. Ridicarea la putere

Operaţia de ridicare la putere a unei matrice este simbolizată cu operatorul (^). Următoarea instrucţiune:

Z = X^p

reprezintă ridicarea la puterea p a matricei X. Expresia X^p are sens doar pentru X matrice pătrată şi p scalar.

Dacă p este un întreg pozitiv, ridicarea la putere este obţinută prin înmulţiri repetate:

$$Z = X^p = X * X * \ldots * X \quad , \forall p > 0$$

iar dacă p este un întreg negativ, X este mai întâi inversată şi apoi se înmulţesc inversele de p ori:

$$Z = X^p = \mathrm{inv}(X) * \mathrm{inv}(X) * \ldots * \mathrm{inv}(X) \quad , \forall p < 0$$

Exemplul 7.4.5. Fie: $Z=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ şi p=-2 . Să se calculeze Z^p.

Cu secvenţa MATLAB:

```
Z=[1 2;3 4]; p=2;
Z1=Z^p
```

se obţine rezultatul:

```
Z1 =
   7  10
  15  22
```

7.4.6. Transpunerea matricelor

Operaţia de transpunere a unei matrice este simbolizată cu operatorul apostrof. Cu instrucţiunea:

Z=Y'

liniile matricei Y devin coloanele matricei transpuse Z. Acest lucru face ca pentru o matrice Y cu dimensiunea m x n să se obţină o matrice Z cu dimensiunea n x m. Dacă elementele matricei Y sunt numere reale, operaţia de transpunere face ca:

Z(i,j)=Y(j,i)

Dacă elementele matricei Y sunt numere complexe, operaţia de transpunere returnează conjugata transpusei, adică:

Z(i,j) = conj(Y(j,i)) = real(Y(j,i))- i*imag(Y(j,i))

Exemplul 7.4.6. Să se determine transpusa matricei $Z=\begin{bmatrix}1 & 1+i\\ i & 3-2*i\end{bmatrix}$.

Cu secvenţa MATLAB:

```
Z=[1 1+i;i  3-2*i];
Z1=Z'
```

se obţin rezultatele:

```
Z1 =
  1.0000                  0 - 1.0000i
  1.0000 - 1.0000i  3.0000 + 2.0000i
```

8. GENERAREA VECTORILOR ŞI A MATRICELOR UZUALE

Funcţiile folosite pentru generarea vectorilor şi a matricelor uzuale sunt:

zeros	Generează matricea nulă;
ones	Generează matricea unitate;
eye	Generează matricea identică;
rand	Generează numere aleatoare cu distribuţie uniformă;
randn	Generează numere aleatoare cu distribuţie normală;
linspace	Generează un vector cu pas liniar;
logspace	Generează un vector cu pas logaritmic;
meshgrid	Generează o matrice a reţelei în planul X-Y;
:	Generează un vector cu pas constant.

8.1. Generarea vectorilor

8.1.1. Generarea vectorilor cu pas liniar

Generarea vectorilor cu pas liniar implică cunoaşterea limitelor intervalului (amin şi amax) şi a pasului dintre două elemente (pas), sau numărul de elemente ale vectorului. Metoda de generare a vectorului se alege funcţie de datele de intrare.

- Dacă se cunosc limitele intervalului (amin şi amax) şi pasul (pas) dintre două elemente, se generează vectorul cu instrucţiunea:

  x = amin : pas : amax

unde: amin, amax şi pas sunt scalari şi pot avea orice valoare reală. Numărul de elemente ale vectorului rezultant x este:

$$N = \left[\frac{amax - amin}{pas}\right] + 1$$

unde „[ ]" semnifică partea întreagă a rezultatului expresiei dintre paranteze. Instrucţiunea presupune că:

- dacă pas > 0, atunci este necesar ca amin < amax,
- dacă pas < 0, atunci este necesar ca amin > amax.

Spre exemplu:

```
x = 2:5:25
x = -20:3:10
x = 5:-2:-4
x =-15:-3:-25
x = 5:15
```

sunt corecte, în timp ce următoarele instrucţiuni:

```
x =  2:-1:5
x = -5:2:-10
```


sunt incorecte.

Dacă pasul se omite, valoarea acestuia este considerată implicit egală cu unitatea.

- Dacă se cunosc limitele intervalului (amin şi amax) şi numărul de elemente (N) ale vectorului generat cu pas liniar, atunci se foloseşte instrucţiunea:

  x = *linspace*(amin, amax, N)

Pasul dintre două elemente rezultă egal cu:

$$pas = \frac{amax - amin}{N - 1}$$

Dacă valoarea N este omisă, atunci aceasta este considerată implicit egală cu 100. Valorile limitelor intervalului, amin şi amax, nu sunt supuse niciunei restricţii şi pot fi date în orice ordine (dacă amin > amax vectorul generat va fi ordonat descrescător).

Exemplul 8.1.1. Să se genereze un vector cu pas liniar, cu limitele: amin =2.5, amax = 7 şi pasul egal cu 1.25. Secvenţa:

```
x = 2.5 : 1.25 : 7
```

conduce la rezultatul:

x = [2.5000 3.7500 5.0000 6.2500]

Exemplul 8.1.2. Să se genereze un vector cu pas liniar, cu limitele : amin = 2.5, amax = 7 şi N = 4 elemente. Secvenţa:

```
x = linspace(2.5, 7, 4)
```

conduce la rezultatul:

x = [2.5000 4.0000 5.5000 7.0000]

Deşi generează acelaşi număr de elemente, prima secvenţă controlează pasul şi poate modifica eventual limita superioară, iar a doua controlează numărul de elemente şi menţine limitele impuse.

8.1.2. Generarea vectorilor cu pas logaritmic

Funcţia *logspace* generează vectori cu pas logaritmic; se apelează cu sintaxa:

x = *logspace*(amin, amax, N)

Vectorul x conţine N elemente distribuite logaritmic între decadele [10^{amin}, 10^{amax}]. Dacă numărul de elemente N este omis, se generează un vector cu 50 elemente distribuite logaritmic între decadele [10^{amin} , 10^{amax}]. Dacă amax=π, elementele vectorului sunt distribuite între 10^{amin} şi π. Valorile limitelor intervalului, amin şi amax, nu au nici o restricţie şi pot fi date în orice ordine. Dacă amin > amax, vectorul generat va fi ordonat descrescător.

Exemplul 8.1.3. Să se genereze un vector cu N=5 elemente distribuite logaritmic pe intervalul [10^{-2} , 10^{2}]. Secvenţa:

```
x = logspace(-2, 2, 5)
```

determină rezultatul:

x = [0.0100 0.1000 1.0000 10.0000 100.0000]

Exemplul 8.1.4. Să se genereze un vector cu N=6 elemente distribuite logaritmic pe intervalul $[10^{-2}, \pi]$. Secvenţa:

```
x = logspace(-2, pi, 6)
```

determină rezultatul:

x = [0.0100 0.0316 0.0997 0.3150 0.9948 3.1416]

8.2. Generarea matricelor

8.2.1. Matricea goală

Deşi în MATLAB nu există instrucţiuni pentru declararea tipurilor de variabile, iar matricele se autodimensionează în timpul utilizării, pentru a creşte viteza de lucru se procedează la crearea unei matrice goale. Acest lucru se face la începutul sesiunii de lucru sau la apelarea unui program. Declararea unei matrice goale se face cu instrucţiunea:

X = [ ]

care asignează lui X matricea de dimensiuni 0x0.

Orice matrice goală trebuie sa aibă cel puţin una din dimensiuni zero. Pentru a testa dacă o matrice X este goală, se foloseşte funcţia MATLAB *isempty*, care se apelează cu sintaxa:

r = *isempty*(X)

şi returnează în variabila r valoarea 1 dacă X este matrice goală şi 0 în caz contrar.

8.2.2. Matricea unitate

Matricea unitate este o matrice cu toate elementele 1:

$$U = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 1 \end{bmatrix}$$

şi poate fi generată cu funcţia *ones*; se apelează cu una dintre sintaxele:

U=*ones*(n) U=*ones*(m,n) U=*ones*(size(A))

unde m şi n sunt scalari, iar A este matrice. Dacă funcţia *ones* este apelată cu un singur argument scalar, matricea generată este o matrice pătrată având dimensiunea argumentului. Apelată cu două argumente scalare (m,n), matricea generată are m linii şi n coloane. Dacă funcţia *ones* are ca argument o matrice A, matricea generată este o matrice unitate de aceleaşi dimensiuni cu matricea A.

Exemplul 8.2.1. Să se genereze o matrice unitate cu dimensiunea 2x2 şi o matrice unitate cu dimensiunea 2x3. Cu secvenţa:

```
A1=ones(2)
A2=ones(2,3)
```


se obţine rezultatul:

```
A1 =                    A2 =
   1   1                    1   1   1
   1   1                    1   1   1
```

Exemplul 8.2.2. Fie matricea: $A = \begin{bmatrix} 1 & 3 & 9 \\ 5 & 7 & 2 \end{bmatrix}$.

Să se genereze o matrice unitate de aceleaşi dimensiuni cu matricea A.
Cu secvenţa:

```
A=[1  3  9;  5  7  2];
U=ones(size(A))
```

se obţine rezultatul:

```
U =
   1   1   1
   1   1   1
```

8.2.3. Matricea zero

Matricea zero este o matrice cu toate elementele zero:

$$O = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

şi poate fi generată cu funcţia *zeros*, care se apelează cu una dintre sintaxele:

O= *zeros*(n) O= *zeros*(m,n) O= *zeros*(size(A))

unde m şi n sunt scalari, iar A este matrice. Dacă funcţia *zeros* este apelată cu un singur argument scalar, matricea generată este o matrice pătrată având dimensiunea argumentului. Apelată cu două argumente scalare (m,n), matricea generată are m linii şi n coloane. Dacă funcţia *zeros* are ca argument o matrice A, matricea generată este o matrice zero de aceleaşi dimensiuni cu matricea A.

8.2.4. Matricea identitate

Matricea identitate este o matrice care are elementele de pe diagonala principală egale cu unu, iar toate celelalte egale cu zero:

$$I = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{bmatrix}$$

Se generează cu funcţia *eye*, care se apelează cu una dintre sintaxele:

I= *eye*(n) I= *eye*(m,n) I= *eye*(size(A))

unde m şi n sunt scalari, iar A este matrice. Dacă funcţia *eye* este apelată cu un singur argument scalar, matricea generată este o matrice pătrată având dimensiunea argumentului. Apelată cu două argumente scalare (m,n), matricea generată are m linii şi n coloane. Dacă funcţia *eye* are ca argument o matrice A, matricea generată este o matrice identitate de aceleaşi dimensiuni cu matricea A.

Exemplul 8.2.3. Să se genereze o matrice identitate cu dimensiunea 2x2 şi o matrice identitate cu dimensiunea 2x3. Cu secvenţa:

```
A1=eye(2); A2=eye(2,3)
```

se obţine rezultatul:

```
A1 =                A2 =
   1   0                1   0   0
   0   1                0   1   0
```

Exemplul 8.2.4. Fie matricea: $A = \begin{bmatrix} 1 & 3 & 9 \\ 5 & 7 & 2 \end{bmatrix}$.

Să se genereze o matrice identitate de aceleaşi dimensiuni cu matricea A.
Cu secvenţa:

```
A=[1  3  9;  5  7  2];
I=eye(size(A))
```

se obţine rezultatul:

```
I =
   1   0   0
   0   1   0
```

8.2.5. Matricea aleatoare

Generarea matricelor cu numere aleatoare se poate face cu:

- funcţia *rand* pentru numere aleatoare cu distribuţie uniformă în intervalul (0.0, 1.0);
- funcţia *randn* pentru numere aleatoare cu distribuţie normală (Gaussiană), de medie zero şi varianţă unu.

Sintaxele pentru generarea matricelor cu numere aleatoare sunt:

Ru=*rand*(n) Rn=*randn*(n)
Ru=*rand*(m,n) Rn=*randn*(m,n)
Ru=*rand*(size(A)) Rn=*randn*(size(A))

unde m şi n sunt scalari, iar A este matrice. Dacă funcţiile *rand* şi *randn* sunt apelate cu un singur argument scalar, matricea generată este o matrice pătrată având dimensiunea argumentului. Apelată cu două argumente scalare (m,n), matricea generată are m linii şi n coloane. Dacă funcţiile *rand* şi *randn* au ca argument o matrice A, matricea generată este o matrice aleatoare de aceleaşi dimensiuni cu matricea A.

Pentru simularea experienţelor care comportă aceleaşi condiţii se generează serii de numere aleatoare la care se controlează un parametru de iniţializare al generatorului, *seed*.
rand('seed') sau *randn*('seed') - returnează valoarea curentă a numărului „seed";
rand('seed',n) sau *randn*('seed',n) - setează numărul „seed" la valoarea n.

Exemplul 8.2.5. Să se genereze o matrice aleatoare cu dimensiunea 2x2 şi o matrice aleatoare cu dimensiunea 2x3 cu elementele distribuite uniform şi normal.
Cu secvenţa:

```
R1u=rand(2)
R1n=rand(2)
R2u=rand(2,3)
R2n=randn(2,3)
```

se obţin rezultatele:

```
R1u =                R2u =
   0.6521  0.6813       0.8456  0.9554  0.1482
   0.1503  0.3858       0.5901  0.5561  0.9833
R1n =                R2n =
   0.3877  0.1475       0.4142 -1.0215  1.5161
   0.4997  0.5872      -0.9778  0.3177  0.7494
```

Exemplul 8.2.6. Fie matricea: $A = \begin{bmatrix} 1 & 3 & 9 \\ 5 & 7 & 2 \end{bmatrix}$.

Să se genereze câte o matrice aleatoare de aceleaşi dimensiuni cu matricea A, cu numerele distribuite uniform şi normal.
Cu secvenţa:

```
A=[1  3  9;  5  7  2];
Ru=rand(size(A))
Rn=randn(size(A))
```

se obţine rezultatul:

```
Ru =                    Rn =
   0.9611  0.1998  0.6293     -0.2091 -1.0639  1.1330
   0.1260  0.3192  0.1267      0.5621  0.3516  0.1500
```

8.3. Generarea unei reţele (mesh)

Funcţia *meshgrid* transformă domeniul specificat prin vectorii x şi y în tablourile X şi Y care pot fi folosite atât pentru evaluarea funcţiilor de două variabile, cât şi pentru reprezentări 3D de tipul *mesh* sau *surface*.
Se apelează cu una dintre sintaxele:

[X, Y]=*meshgrid*(x, y) [X, Y]=*meshgrid*(x)

şi returnează datele în tablourile X şi Y, care sunt sunt copii ale vectorului x şi ale vectorului y. Cu alte cuvinte, funcţia *meshgrid* returnează în tablourile X şi Y perechile de coordonate

ale tuturor punctelor din domeniul definit de vectorii x şi y. Pentru exemplificare vezi figura 8.1.

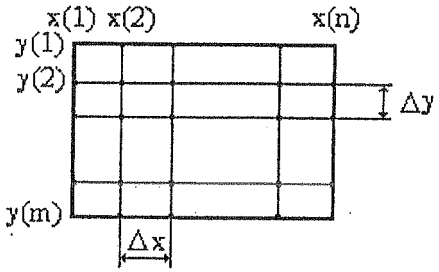


Fig. 8.1. Mesh-ul rezultat prin divizarea domeniului xy.

Exemplul 8.3.1. Să se genereze tablourile X şi Y pentru domeniul:

$$-2 \le x \le 2, \quad -2 \le y \le 3$$

cu pasul 2 pe axa Ox şi pasul 1 pe axa Oy.
Cu instrucţiunea:

```
[X, Y]=meshgrid (-2: 2:2, -2: 2)
```

se obţine rezultatul:

```
X =                    Y =
  -2   0   2             -2  -2  -2
  -2   0   2             -1  -1  -1
  -2   0   2              0   0   0
  -2   0   2              1   1   1
  -2   0   2              2   2   2
```


9. FUNCŢII MATEMATICE UZUALE

9.1. Aproximarea numerelor

Funcţiile MATLAB folosite pentru aproximarea numerelor sunt:

ceil Returnează un număr întreg, rotunjit la cel mai apropiat întreg spre plus infinit (+∞);
fix Returnează un număr întreg, rotunjit la cel mai apropiat întreg spre zero;
floor Returnează un număr întreg, rotunjit la cel mai apropiat întreg spre minus infinit (-∞);
round Returnează un număr întreg, rotunjit la cel mai apropiat întreg;
rem Returnează restul împărţirii argumentelor;
rat Returnează aproximarea unui număr cu fracţii raţionale continue;
rats Returnează aproximarea unui număr cu numere raţionale;
sign Returnează semnul argumentului.

9.1.1. Aproximarea cu numere întregi

Funcţiile MATLAB folosite pentru aproximarea cu numere întregi sunt: *fix, floor, ceil, round.* Aceste funcţii operează asupra fiecărui element al unei matrice sau al unui vector şi se apelează cu sintaxa:

nume_funcţie(argument)

unde:

- nume_funcţie este numele uneia dintre funcţiile de mai sus, iar
- argument poate fi un scalar, un vector sau o matrice, ale căror elemente se doresc a fi rotunjite.

Dacă elementele din argumentul funcţiei sunt numere complexe, funcţiile de mai sus operează independent asupra fiecărei părţi (reală şi imaginară).

Exemplul 9.1.1. Să se rotunjească elementele vectorului:

V = [0 2 2.3 4.7 -5.2 -7.8]

- la cel mai apropiat întreg;
- la cel mai apropiat întreg spre 0;
- la cel mai apropiat întreg spre +∞;
- la cel mai apropiat întreg spre -∞.

Să se determine semnul elementelor vectorului V.
Cu secvenţa de instrucţiuni:

```
V = [ 0  2  2.3  4.7  -5.2  -7.8 ]
A = round(V)
B = fix(V)
C = ceil(V)
D = floor(V)
E = sign(V)
```

se obţin rezultatele:

```
A = [ 0  2  2  5  -5  -8 ]
B = [ 0  2  2  4  -5  -7 ]
C = [ 0  2  3  5  -5  -7 ]
D = [ 0  2  2  4  -6  -8 ]
E = [ 0  1  1  1  -1  -1 ]
```

Exemplul 9.1.2. Acelaşi enunţ ca mai sus, pentru o matrice cu elemente numere complexe:

$$V=\begin{bmatrix} 1.25+2.59i & 7.3-5.3i \\ -4.2+1.8i & -2.6-1.4i \end{bmatrix}$$

Cu secvenţa de instrucţiuni:

```
V=[1.25+2.59*i   7.3-5.3*i;
  -4.2+1.8*i    -2.6-1.4*i]
A = round(V)
B = fix(V)
C = ceil(V)
D = floor(V)
E = sign(V)
```

se obţin rezultatele:

```
A =
  1.0000 + 3.0000i  7.0000 - 5.0000i
 -4.0000 + 2.0000i -3.0000 - 1.0000i
B =
  1.0000 + 2.0000i  7.0000 - 5.0000i
 -4.0000 + 1.0000i -2.0000 - 1.0000i
C =
  2.0000 + 3.0000i  8.0000 - 5.0000i
 -4.0000 + 2.0000i -2.0000 - 1.0000i
D =
  1.0000 + 2.0000i  7.0000 - 6.0000i
 -5.0000 + 1.0000i -3.0000 - 2.0000i
```

9.1.2. Aproximarea cu numere raţionale

Funcţia MATLAB *rats* realizează aproximarea cu numere raţionale; se apelează cu una dintre sintaxele:

y=*rats*(x) y=*rats*(x, 's')

Argumentul de intrare 's' determină afişarea rezultatului simbolic y într-o matrice şir.

Exemplul 9.1.3. Să se aproximeze cu numere raţionale, numerele: 1.25, 0.25, π şi 1.2596.

Cu secvenţa:

```
X=[1.25 0.25 pi 1.2596];
Y=rats(X)
```

se obţin rezultatele:


```
Y = [ 5/4  1/4  355/113  820/651 ]
```

9.1.3. Aproximarea cu fracţii continue

Funcţia MATLAB *rat* aproximează un număr cu fracţii continue; se apelează cu una dintre sintaxele:

y=*rat*(x) [a, b]=*rat*(x)
y=*rat*(x, tol) [a, b]=*rat*(x, tol)

unde:

- x - este numărul care trebuie aproximat cu fracţii continue;
- tol - este toleranţa care se acceptă între numărul x şi numărul y (y-x<=tol); implicit, tol=10^{-6};
- y - exprimarea lui x ca fracţie continuă;
- a şi b - numărătorul şi numitorul fracţiei care aproximează pe x cu toleranţa tol.

Funcţia *rat* aproximează fiecare element al vectorului x cu un număr de forma:

$$y=d_0+\cfrac{1}{d_1+\cfrac{1}{d_2+\ldots+\cfrac{1}{d_k}}}$$

Exemplul 9.1.4. Să se aproximeze cu fracţii continue numerele: 0.25, 1.25, -2.25 şi 1.343. Cu secvenţa MATLAB

```
rat([0.25 1.25 -2.25 1.343])
```

se obţine rezultatul:

```
0 + 1/(4)
1 + 1/(4)
-2 + 1/(-4)
1 + 1/(3 + 1/(-12 + 1/(6)))
```

Soluţia corespunde următoarelor descompuneri în fracţii continue:

$$0.25=\frac{1}{4} \qquad 1.25=1+\frac{1}{4}$$

$$-2.25=-2+\frac{1}{-4} \qquad 1.343=1+\cfrac{1}{3+\cfrac{1}{-12+\cfrac{1}{6}}}$$

Exemplul 9.1.5. Să se aproximeze prin fracţii raţionale numerele: 2.25, 3.5, 6.57, 10. Se înscriu aceste numere într-un vector şi se aplică funcţia *rat*, ca în secvenţa de mai jos:

```
X = [2.25 3.5 6.57 10 ]
[A,B] =rat(X)
```

obţinându-se rezultatul:

```
X = [ 2.25  3.5   6.57   10 ]
A = [ 9     7     657    10 ]
```

B = [4 2 100 1]

adică o aproximare a numerelor date, prin următoarele fracţii:

2.25 = 9/4 3.5 = 7/2

6.57 = 657/100 10 = 10/1

9.1.4. Funcţia rest

Funcţia *rem*(X,Y) calculează restul împărţirii lui X la Y, element cu element. Dacă elementele vectorului sau matricei sunt numere complexe, partea imaginară este ignorată. Argumentele X şi Y trebuie să fie matrice de aceeaşi dimensiune, sau unul dintre ele să fie scalar.

Exemplul 9.1.6. Să se calculeze restul împărţirii elementelor vectorului X la Y.
Cu secvenţa de instrucţiuni:

```
X = [ 1  3 -6 ];
Y = [ 2  3  4 ];
Z = rem(X,Y)
```

se obţine rezultatul:

Z = [1 0 -2]

Exemplul 9.1.7. Să se determine restul împărţirii unui vector la un scalar şi al unui scalar la un vector.
Cu secvenţa MATLAB:

```
X= [2.5  6  -7];
Y= 3;
Z1=rem(X,Y)
Z2=rem(Y,X)
```

se obţin rezultatele:

Z1 = [2.5000 0 -1.0000]

Z2 = [0.5000 3.0000 3.0000]

Exemplul 9.1.8. Fie vectorii:

X= [1 i -6+2i]

Y= [2+i 3+2i 3-i]

Să se calculeze restul împărţirii elementelor complexe ale vectorului X la Y.
Cu secvenţa de instrucţiuni:

```
X = [1  i -6+2*i];
Y = [2+i 3+2*i 3-i];
Z = rem(X,Y)
```

se obţine rezultatul:

Z = [1 0 0]

9.1.5. Funcţia semn

Funcţia *sign* asociază fiecărui element al vectorului X elementele -1, 0, 1, după următoarea regulă:

$$\text{sign}(X) = \begin{cases} 1, & \text{dacă } X>0 \\ 0, & \text{dacă } X=0 \\ -1, & \text{dacă } X<0 \end{cases}$$

9.2. Divizori şi multipli comuni

Pentru calculul divizorilor şi al multiplilor comuni se folosesc funcţiile:

gcd Calculează cel mai mare divizor comun a două numere;

lcm Calculează cel mai mic multiplu comun a două numere.

9.2.1. Cel mai mare divizor comun

Funcţia MATLAB *gcd* calculează cel mai mare divizor comun a două numere întregi; se apelează cu sintaxa:

a=*gcd*(x,y)

Exemplul 9.2.1. Să se determine cel mai mare divizor comun al numerelor: 30 şi 21. Cu secvenţa:

```
a=gcd(30,21)
```

rezultă: a = 3

9.2.2. Cel mai mic multiplu comun

Funcţia MATLAB *lcm* returnează cel mai mic multiplu comun a două numere întregi. Se apelează cu sintaxa:

a=*lcm*(x,y)

Exemplul 9.2.2. Să se determine cel mai mic multiplu comun al numerelor : 9 şi 30.
Cu secvenţa:

```
a=lcm(9,30)
```

rezultă:

a = 90

9.3. Numere complexe

Operaţiile cu numere complexe folosesc funcţiile:

abs Calculează valoarea absolută (modulul) a argumentului;

angle Calculează faza (unghiul) argumentului;

unwrap Calculează părţile reală şi imaginară ale numerelor complexe exprimate în forma polară;

conj Calculează conjugata complexă a argumentului;

imag Extrage partea imaginară a argumentului;

real Extrage partea reală a argumentului.

9.3.1. Definirea numerelor complexe în MATLAB

Numerele complexe sunt permise în toate operaţiile şi funcţiile din MATLAB. Acestea sunt introduse utilizând variabilele speciale i şi j, ca în exemplele:

z = 3 + 4*i

z = 3 + 4*j

Pentru a defini matricea :

$$M=\begin{bmatrix} 2+3i & i \\ -1 & -5+i \end{bmatrix}$$

există două metode:

- ca sumă a două matrice cu elemente numere reale, una reprezentând partea reală iar cealaltă partea imaginară;

  M=[2 0; -1 -5] +[3 1 0 1]*i

- ca o matrice cu elemente numere complexe.

  M=[2+3*i i; -1 -5+i]

Prin al doilea procedeu trebuie evitat orice spaţiu liber (blanc) între părţile reală şi imaginară ale aceluiaşi număr complex; astfel:

a=[2+3*i]

b=[2 +3*i]

returnează:

a = [2.0000+ 3.0000i]

b = [2.0000 0 + 3.0000i]

notaţia b reprezentând două numere separate.

Dacă variabilele i sau j au fost deja utilizate în alte scopuri, pentru calculul cu numere complexe poate fi declarată o nouă unitate imaginară, în modul următor:

ii = sqrt(-1)

9.3.2. Modulul şi argumentul numerelor complexe

Un număr complex z se exprimă sub una dintre formele:

- carteziană: z=x+iy
- polară: $z=re^{i\varphi}$

unde x şi y sunt părţile reală şi imaginară ale numărului complex z, iar r şi φ sunt modulul şi argumentul numărului complex z.

Funcţia *abs* determină modulul elementelor unui vector sau unei matrice; se apelează cu sintaxa:

r=abs(z)

Funcţia *angle* calculează argumentul elementelor unui vector sau unei matrice, în radiani; se apelează cu sintaxa:

fi=*angle*(z)

Funcţia *unwrap* permite calculul părţilor reale, cu sintaxa:

x= *unwrap*(z) sau

x=*unwrap*(*real*(z))

respectiv a părţii imaginare, cu sintaxa:

y= *unwrap*(*imag*(z))

Argumentul numărului complex trebuie să fie exprimat în radiani.

Exemplul 9.3.1. Să se scrie numărul complex z=1+i sub forma polară.

Cu instrucţiunile:

```
x=1; y=1;
z=x+i*y;
r=abs(z)
fi=angle(z)
```

se obţine rezultatul:

r = 1.4142

fi = 0.7854

Exemplul 9.3.2. Să se scrie numărul complex $z=4e^{i\pi/4}$ sub forma carteziană.

Cu instrucţiunile:

```
r = sqrt(4);
f = pi/4;
z = r.*exp(i*fi)
```

se obţine rezultatul:

z = 1.4142 + 1.4142i

Exemplul 9.3.3. Fie dată matricea cu numerele complexe exprimate sub forma polară:

$$M=\begin{bmatrix} 1 & e^{i\frac{\pi}{4}} & e^{-i\frac{\pi}{4}} \\ e^{i(\frac{\pi}{2}+\frac{\pi}{4})} & e^{i(2\pi+\frac{\pi}{4})} & e^{-i(2\pi+\frac{\pi}{4})} \end{bmatrix}$$

Să se determine proiecţiile acestor numere pe axele reală şi imaginară, aplicând funcţia *unwrap*

Cu instrucţiunile:

```
M=[1, exp(i*pi/4), exp(-i*pi/4);
exp(i*(pi/2+pi/4)), exp(i*(2*pi+pi/4)),...
exp(-i*(2*pi+ pi/4))];
x=unwrap(real(M))
y=unwrap(imag(M))
```

se obţine:

```
x =
    1.0000    0.7071    0.7071
   -0.7071    0.7071    0.7071
y =
         0    0.7071   -0.7071
    0.7071    0.7071   -0.7071
```

9.3.3. Părţile reală şi imaginară şi conjugatul numerelor complexe

Partea reală a unui număr complex poate fi determinată cu funcţia *real*; se apelează cu sintaxa:

x=*real*(z)

iar partea imaginară poate fi determinată cu funcţia *imag* ; se apelează cu sintaxa:

y=*imag*(z)

Conjugatul $\overline{z}$ al unui număr complex se poate determina cu funcţia *conj* ; se apelează cu sintaxa:

conj=*conj*(z)

Exemplul 9.3.4. Fie dată matricea $M=\begin{bmatrix} 1 & 2+i \\ -i & 4e^{i\frac{\pi}{4}} \end{bmatrix}$

Să se determine partea reală, imaginară şi conjugatul elementelor acesteia.

Cu secvenţa MATLAB:

```
M = [ 1  2+i; -i  4*exp(i*pi/4) ];
Mr=real(M)     % partea reala
Mi=imag(M)     % partea imaginara
Mc=conj(M)     % matricea conjugata
```

se obţin rezultatele :

```
Mr =
        1.0000   2.0000
             0   2.8284
Mi =
             0   1.0000
       -1.0000   2.8284
Mc =
        1.0000          2.0000 - 1.0000i
        0 + 1.0000i     2.8284 - 2.8284i
```

9.4. Funcţiile putere, radical, logaritm şi exponenţială

Funcţiile MATLAB pentru ridicarea la putere, extragerea radicalului, calculul logaritmului şi al exponenţialei, sunt:

^	Ridică un număr a la puterea n (a^n);
exp	Calculează exponenţiala (e^x);
log	Calculează logaritmul natural (ln);
log2	Calculează logaritmul în baza 2 ($\log_2$);
log10	Calculează logaritmul zecimal ($\log_{10}$);
nextpow2	Determină puterea N a numărului 2 care majorează modulul argumentului P ($\lvert P\rvert \le 2^N$);
pow2	Calculează valoarea numărului 2 la puterea n (2^n);
sqrt	Calculează radicalul de ordinul doi dintr-un număr.

Dacă argumentul acestor funcţii elementare sunt matrice, ele operează element cu element. Argumentele funcţiilor pof fi şi numere complexe.

9.4.1. Funcţia putere

MATLAB-ul dispune de două funcţii pentru ridicarea la putere:

- *pow2* - pentru a ridica 2 la puterea n (2^n),
- ^ - pentru a ridica un număr a la puterea n ($x=a^n$).

Se apelează cu sintaxele:

y=*pow2*(x) - calculează numărul $y=2^x$. Dacă x este o matrice, y va fi o matrice de aceleaşi dimensiuni cu elementele calculate după această regulă, funcţia acţionând element cu element.

y=*pow2*(m,n) - calculează numărul $y=m*2^n$;

x=a^n - calculează puterea n a numărului a, $x=a^n$. Exponentul n poate avea orice valoare, reală sau complexă. Pentru calculul radicalului de ordinul n dintr-un număr a, se utilizează funcţia putere sub forma: $x=a^{1/n}$.

Funcţia *nextpow2* având ca argument scalarul P, se apelează cu sintaxa:

N=*nextpow2*(P)

şi returnează cel mai mic număr natural N astfel încât $2^N \ge |P|$.

Dacă P este vector, funcţia returnează scalarul N, astfel încât 2^N majorează numărul de elemente ale vectorului.

Exemplul 9.4.1. Să se calculeze: $A=\begin{bmatrix} 2^3 & 2^5 & 2^{13.5} \end{bmatrix}$.

Cu secvenţa:

```
A=[pow2(3)   pow2(5)   pow2(13.5)]
```

rezultă:

```
A =
 1.0e+004 *
   0.0008   0.0032   1.1585
```

adică:

```
A = [ 8   32   11585 ]
```

Exemplul 9.4.2. Să se efectueze aceleaşi calcule ca la punctul anterior, utilizând operatorul de ridicare la putere ^.

Cu secvenţa:

```
A=[2^3   2^5   2^13.5]
```

se obţin aceleaşi rezultate.

Exemplul 9.4.3. Să se calculeze: $x=\sqrt[3]{125}$

Cu secvenţa:

```
x=125^(1/3)
```

se obţine:

```
x =    5.0000
```

Exemplul 9.4.4. Să se determine puterile N ale lui 2 care majorează elementele vectorului $A=\begin{bmatrix} 4 & -8 & 17 \end{bmatrix}$. Să se calculeze vectorul majorant $P=2^N$.

Cu secvenţa:

```
N=[nextpow2(4) nextpow2(-8) nextpow2(17)]
```

```
P=2.^N
```

se obţin rezultatele:

N = [2 3 5]
P = [4 8 32]

9.4.2. Funcţia radical

Calculul radicalului de ordinul 2 dintr-un număr, $x = \sqrt{a}$, poate utiliza funcţia putere, sau funcţia *sqrt*, apelată cu sintaxa:

x=*sqrt*(a)

Argumentul a poate fi orice număr real sau complex. Dacă numărul a este negativ sau complex, rezultatul calculului este un număr complex.

Exemplul 9.4.5. Să se calculeze radicalul fiecărui element al matricei: $X = \begin{bmatrix} 1 & 2 \\ 4 & -9 \end{bmatrix}$.

Cu secvenţa de instrucţiuni:

```
X = [ 1 2; 4 -9 ]
c = sqrt(X)
```

se obţine rezultatul:

```
c =
    1.0000        1.4142
    2.0000        0 + 3.0000i
```

9.4.3. Funcţia logaritm

Calculul logaritmului natural al logaritmului în baza 2 sau al logaritmului în baza 10 al unui număr a utilizează funcţiile *log*, *log2* şi respectiv *log10*, apelate cu sintaxele:

x=*log*(a) x=*log2*(a) x=*log10*(a)

Dacă argumentul funcţiilor *log* şi *log10* este un număr negativ, sau complex, z=x+iy, rezultatul este calculat cu relaţiile:

log(z) = *log*(*abs*(z)) + i *atan2*(y,x)
log10(z) = *log10*(*abs*(z)) + i *atan2*(y,x)

unde *atan2* este funcţie MATLAB ce calculează arctangenta numărului complex.

Exemplul 9.4.6. Să se calculeze logaritmul natural şi zecimal din numerele e^2 şi 100.

Cu secvenţa:

```
x1=[ log(exp(2)) log(100)]
x2=[ log10(exp(2)) log10(100)]
```

sau:

```
A=[ exp(2) 100]
x1=log(A)
x2=log10(A)
```

se obţine rezultatul:

x1 = [2.0000 4.6052]
x2 = [0.8686 2.0000]

Exemplul 9.4.7. Să se calculeze logaritmul în bază 2 al elementelor matricei:

$A = \begin{bmatrix} 4 & 2^3 & 8^2 & 10 \end{bmatrix}$.

Cu secvenţa:

```
A=[4  2^3 ;  8^2   10];
x=log2(A)
```

se obţine rezultatul:

```
x =
    2.0000    3.0000
    6.0000    3.3219
```

9.4.4. Funcţia exponenţială

Calculul exponenţialei: $x=e^a$, (unde e= 2.71828182845...) foloseşte funcţia *exp*, apelată cu sintaxa:

x=*exp*(a)

Dacă argumentul este numărul complex z=x+iy, rezultatul este calculat cu relaţia:

$e^z = e^x (\cos(y) + i \sin(y))$

Exemplul 9.4.8. Să se calculeze: e, e^2 şi e^{-3}.

Cu secvenţa:

```
x=[exp(1)   exp(2)   exp(-3)]
```

sau:

```
A=[1  2  -3]
x=exp(A)
```

se obţine rezultatul:

x = [2.7183 7.3891 0.0498]

9.5. Funcţiile trigonometrice

Funcţiile trigonometrice se apelează cu sintaxa:

x=nume_funcţie(argument)

unde:

- nume_funcţie este numele uneia dintre funcţiile trigonometrice de mai jos;
- argument este valoarea pentru care se evaluează funcţia;
- x este variabila în care se returnează rezultatul.

Dacă argumentul este o matrice, funcţiile trigonometrice operează asupra fiecărui element.

9.5.1. Funcţiile trigonometrice directe

Funcţiile trigonometrice directe în MATLAB sunt:

sin Calculează sinusul argumentului;
cos Calculează cosinusul argumentului;

tan Calculează tangenta argumentului;
cot Calculează cotangenta argumentului;
sec Calculează secanta argumentului;
csc Calculează cosecanta argumentului.

Pentru argumente numere complexe, z=x+iy, relaţiile de calcul sunt:

$$sin(z) = sin(x)cosh(y) + i\ cos(x)sinh(y)$$
$$cos(z) = cos(x)cosh(y) - i\ sin(x)sinh(y)$$
$$tan(z) = sin(z)/cos(z)$$

Exemplul 9.5.1. Să se calculeze funcţiile trigonometrice directe ale elementelor vectorului:

$$X=\left[\frac{\pi}{4}\quad \frac{3\pi}{4}\quad 2\pi-\frac{\pi}{4}\right]$$

Cu secvenţa:

```
X=[ pi/4   3*pi/4    2*pi-pi/4];
A=sin(X)
B=tan(X)
```

se obţin rezultatele:

```
A = [ 0.7071   0.7071  -0.7071 ]
B = [ 1.0000  -1.0000  -1.0000 ]
```

9.5.2. Funcţiile trigonometrice inverse

Funcţiile trigonometrice inverse în MATLAB sunt:

asin Calculează arcsinusul argumentului;
acos Calculează arccosinusul argumentului;
atan Calculează arctangenta argumentului;
atan2 Calculează arctangenta unui argument complex;
acot Calculează arccotangenta argumentului;
asec Calculează arcsecanta argumentului;
acsc Calculează arccosecanta argumentului.

Exemplul 9.5.2. Să se calculeze funcţiile trigonometrice inverse pentru elementele vectorului:

$$X=\left[0\quad 1\quad \sqrt{2}/2\right]$$

Cu secvenţa:

```
X=[0   1   sqrt(2)/2];
A=asin(X)
B=atan(X)
```

se obţin rezultatele:

```
A = [   0        1.5708     0.7854 ]
B = [   0        0.7854     0.6155 ]
```


9.6. Funcţiile hiperbolice

Funcţiile hiperbolice se apelează cu sintaxa:

x=nume_funcţie(argument)

unde:

- nume_funcţie este numele uneia dintre funcţiile hiperbolice de mai jos;
- argument este valoarea pentru care se evaluează funcţia;
- x este variabila în care se returnează rezultatul.

Dacă argumentul este o matrice, funcţiile trigonometrice operează asupra fiecărui element.

9.6.1. Funcţiile hiperbolice directe

Funcţiile hiperbolice directe în MATLAB sunt:

sinh Calculează sinusul hiperbolic al argumentului.
cosh Calculează cosinusul hiperbolic al argumentului.
tanh Calculează tangenta hiperbolică a argumentului.
coth Calculează cotangenta hiperbolică a argumentului.
sech Calculează secanta hiperbolică a argumentului.
csch Calculează cosecanta hiperbolică a argumentului.

9.6.2. Funcţiile hiperbolice inverse

Funcţiile hiperbolice inverse în MATLAB sunt:

asinh Calculează arcsinusul hiperbolic al argumentului;
acosh Calculează arccosinusul hiperbolic al argumentului;
atanh Calculează arctangenta hiperbolică a argumentului;
acoth Calculează arccotangenta hiperbolică a argumentului;
asech Calculează arcsecanta hiperbolică a argumentului;
acsch Calculează arccosecanta hiperbolică a argumentului.

10. MATRICE SPECIALE

În MATLAB există următorul set de funcţii care generează matrice speciale, folosite în aplicaţii de algebră liniară şi prelucrarea semnalelor:

compan	Calculează companionul matriceal;
diag	Generează o matrice diagonală;
hadamard	Generează matricea Hadamard;
hankel	Generează matricea Hankel;
hilb	Generează matricea Hilbert;
invhilb	Generează matricea Hilbert inversă;
kron	Calculează tensorul Kronecker;
magic	Generează pătratul magic;
pascal	Generează matricea Pascal;
toeplitz	Generează matricea Toeplitz;
vander	Generează matricea Vandermonde;
wilkinson	Generează matricea Wilkinson.

10.1. Companionul matriceal

Funcţia *compan* calculează companionul unei matrice; se apelează cu sintaxa:

A=*compan*(p)

Dacă p este un vector cu coeficienţii polinomiali scrişi în ordine descrescătoare a puterilor, *compan(p)* este companionul matriceal corespondent. Prima linie a companionului este constituită din numere reprezentând câtul cu semn schimbat dintre coeficienţii polinomiali, începând cu puterea (n-1) până la termenul liber şi coeficientul polinomial al puterii n, adică -p(2:n)/p(1). Pentru polinomul:

$$P(x) = p(1)x^n + p(2)x^{n-1} + p(3)x^{n-2} + \ldots + p(n-1)x^2 + p(n)x^1 + p(n+1)$$

prima linie a companionului este:

-p(2)/p(1), -p(3)/p(1) ..., -p(n+1)/p(1).

Valorile proprii ale companionului unui vector p, ai cărui coeficienţi polinomiali sunt daţi de funcţia *eig(compan(p))*, sunt rădăcinile polinomului.

Prin urmare, pentru a determina rădăcinile unui polinom, se poate folosi expresia MATLAB:

r=*eig*(*compan*(p))

unde p sunt coeficienţii polinomului în ordinea descrescătoare a puterilor.

Exemplul 10.1.1. Să se determine rădăcinile polinomului:

$p(x)=x^3 - x^2 - x + 1$

Se scrie vectorul coeficienţilor polinomiali, se determină companionul şi apoi valorile proprii ale acestuia. Valorile proprii ale companionului sunt chiar rădăcinile polinomului.

Cu secvenţa de instrucţiuni MATLAB:

```
p = [ 1 -1 -1 1 ];
b = compan(p);
```



```
r = eig(b)
```

sau, mai compact:

```
r=eig(compan(p))
```

se obţin rădăcinile polinomului:

```
x = [ 1.0000   1.0000  -1.0000 ]
```

10.2. Matricea diagonală

Funcţia *diag* generează matrice diagonale sau operează cu diagonalele unor matrice; se apelează cu una dintre sintaxele:

D=*diag*(X) D=*diag*(X,k)

Argumentul opţional k indică diagonala corespunzătoare, anume:

k = 0, diagonala principală;

k > 0, indică diagonala k de deasupra celei principale;

k < 0, indică diagonala k de sub cea principală.

Dacă argumentul X este un vector cu n componente, funcţia *diag*(X,k) generează o matrice pătrată de ordinul n+abs(k), cu elementele vectorului X pe diagonala k. Apelată fără argumentul k, funcţia creează o matrice cu elementele vectorului X pe diagonala principală.

Dacă argumentul X este o matrice, funcţia *diag*(X,k) extrage un vector coloană format din elementele de pe diagonala k a matricei X.

Exemplul 10.2.1. Fie dată matricea $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Să se genereze o matrice având diagonala principală identică cu cea a matricei A şi cu celelalte elemente egale cu zero. Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 9];
B=diag(diag(A))
```

se obţine rezultatul:

```
B =
   1   0   0
   0   5   0
   0   0   9
```

Exemplul 10.2.2. Să se genereze o matrice de ordinul 2m+1, definită de relaţiile:

$$M(i,j) = \begin{cases} i-m-1 & , \ i=j \\ 1 & , \ i=j\pm 1 \\ 0 & , \ \hat{\imath}\text{n rest} \end{cases}$$

unde $i \in [1, 2m+1]$ şi $j \in [1, 2m+1]$.

Pentru m=2, matricea rezultat este: $M=\begin{bmatrix} -2 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$

Pentru a genera matricea tridiagonală de ordinul 2m+1, din acest exemplu, se utilizează secvenţa MATLAB:

```
m= 2
M=diag(-m:m)+diag(ones(2*m,1),1)+...
      diag(ones(2*m,1),-1)
```

Exemplul 10.2.3. Să se aplice funcţia *diag* vectorului V = [1 2 3].
Cu secvenţa MATLAB:

V = [1 2 3]
A = *diag*(V,2)
B = *diag*(V)

se obţin rezultatele:

A =

0	0	1	0	0
0	0	0	2	0
0	0	0	0	3
0	0	0	0	0
0	0	0	0	0

B =

1	0	0
0	2	0
0	0	3

10.3. Matricea Hadamard

Matricea Hadamard este o matrice pătrată cu toate elementele 1 sau -1 aşezate într-o astfel de succesiune, încât linia j este identică cu coloana j. Funcţia *hadamard* se apelează cu sintaxa:

Hn=*hadamard*(n)

unde ordinul n al matricei trebuie să fie astfel ales, încât n, n/12 sau n/20 să fie o putere a lui 2. Această matrice nu este însă ortogonală (motiv pentru care s-a folosit notaţia Hn), dar se poate ortogonaliza prin înmulţirea cu un scalar, conform relaţiei:

$$[H_{orth.n}] = \frac{1}{\sqrt{n}}[H_n] \quad \text{unde } n=2^k\text{, k fiind un număr natural}$$

Exemplul 10.3.1. Să se traseze liniile de nivel corespunzătoare matricei Hadamard pentru n=12 şi n=40. Cu secvenţa MATLAB:

```
n=12; contour(hadamard(n))
```

se pot urmări liniile de nivel în fereastra grafică.

Exemplul 10.3.2. Să se determine matricele Hadamard ortogonală şi neortogonală, de ordinul 2.
Cu instrucţiunile:


```
H2n=hadamard(2)
H2o=(1/sqrt(2))*H2n
```

se obţine rezultatul:

H2n =

1	1
1	-1

H2o =

0.7071	0.7071
0.7071	-0.7071

10.4. Matricea Hankel

Matricea Hankel este o matrice anti-simetrică şi anti-diagonală; se generează cu funcţia *hankel* apelată cu una dintre sintaxele:

H = *hankel*(c) H = *hankel*(c,l)

Dacă funcţia *hankel* este apelată cu un singur vector argument, matricea returnată este o matrice Hankel pătrată, a cărei primă coloană şi primă linie sunt elementele vectorului argument; toate elementele de sub prima anti-diagonală sunt nule. În schimb, dacă funcţia *hankel* este apelată cu doi vectori argument, de dimensiuni m şi n, matricea returnată este o matrice Hankel cu dimensiunea mxn. Prima coloană este definită de primul vector argument, iar ultima linie - de elementele celui de-al doilea vector argument. Dacă ultimul element al primului vector (prima coloană) diferă de primul element al celui de-al doilea vector (ultima linie), elementul coloanei are prioritate faţă de elementul liniei.

Anti-diagonalele paralele cu anti-diagonala principală au acelaşi element. De exemplu:

$$H=\begin{bmatrix} 5 & 4 & 1 & 2 \\ 4 & 1 & 2 & 0 \\ 1 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

este o matrice Hankel definită prin vectorul c=[5 4 1 2].

Exemplul 10.4.1. Să se genereze o matrice Hankel, definită de vectorul c=[1 2 3 4].
Cu secvenţa MATLAB:

```
c = [1 2 3 4]
H = hankel(c)
```

se obţine rezultatul:

H =

1	2	3	4
2	3	4	0
3	4	0	0
4	0	0	0

Exemplul 10.4.2. Să se genereze o matrice Hankel, definită de vectorii: c=[10 5 6] (prima coloană) şi l=[1 2 3 4] (ultima linie).
Cu secvenţa MATLAB:

```
c=[10 5 6 ]; l=[1 2 3 4];
H = hankel(c,l)
```

rezultă:

```
H =
   10   5   6   2
    5   6   2   3
    6   2   3   4
```

Se observă prioritatea ultimului element al vectorului, care defineşte prima coloană, faţă de primul element al vectorului, care defineşte ultima linie.

10.5. Matricea Hilbert

Matricea Hilbert este o matrice slab condiţionată, ale cărei elemente sunt definite de relaţia:

$$H(i,j)=\frac{1}{i+j-1}$$

Funcţia *hilb* se apelează cu sintaxa:

H=*hilb(n)*

Exemplul 10.5.1. Să se determine matricea Hilbert de ordinul 3.
Cu instrucţiunea MATLAB:

```
H = hilb(3)
```

se obţine rezultatul:

```
H =
   1.0000   0.5000   0.3333
   0.5000   0.3333   0.2500
   0.3333   0.2500   0.2000
```

10.6. Inversa matricei Hilbert

Funcţia *invhilb* returnează inversa matricei Hilbert de ordinul n; se apelează cu sintaxa:

IH = *invhilb*(n)

Rezultatele sunt exacte pentru n ≤ 15. Compararea cu rezultatul obţinut din calculul *inv*(*hilb*(n)) conduce la observaţia că pentru n > 15 apar erori inacceptabile, ca urmare a slabei condiţionări a matricelor Hilbert.

Exemplul 10.6.1. Calculaţi inversa matricelor Hilbert de ordinele 3 şi 25, atât cu funcţia *invhilb* cât şi prin inversarea matricei H cu funcţia *inv*. Urmăriţi diferenţele rezultatelor. Cu secvenţa MATLAB:

```
n=3
IH1=invhilb(n);
IH2=inv(hilb(n));
D=IH1-IH2;
dmax=max(max(abs(D)))
```

rezultă cea mai mare diferenţă:

dmax = 7.3896e-013

pentru n=3, şi:

dmax = 1.3872e+035

pentru n=25. Rezultatul confirmă existenţa unui algoritm specific implementat în funcţia *invhilb* şi atrage atenţia asupra erorilor ce pot rezulta prin utilizarea unei funcţii fără cunoaşterea particularităţilor acesteia.

10.7. Tensorul Kroneker

Funcţia K=*kron*(X,Y) calculează produsul tensorial Kronecker dintre matricele X şi Y. Rezultatul este o matrice formată din toate produsele posibile între elementele lui X şi Y. Dacă X este o matrice m x n, iar Y este o matrice l x p, atunci:

K=*kron*(X,Y) este o matrice cu dimensiunea ml x np.

De reţinut că produsul tensorial nu este comutativ:

kron(X,Y) ≠ *kron*(Y,X)

De exemplu, dacă X este o matrice 2x3:

X=[X(1,1) X(1,2) X(1,3)
X(2,1) X(2,2) X(2,3)]

atunci K=*kron*(X,Y) are ca rezultat matricea:

K= [X(1,1)*Y X(1,2)*Y X(1,3)*Y
X(2,1)*Y X(2,2)*Y X(2,3)*Y]

Exemplul 10.7.1. Fiind date matricele: $X=\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix}$, $Y=\begin{bmatrix}1&2\\3&4\end{bmatrix}$.

să se calculeze produsele tensoriale Kroneker: XY şi YX.
Cu secvenţa MATLAB:

```
X=[1 2; 3 4; 5 6];
Y=[1 2; 3 4];
K1=kron(X,Y)
K2=kron(Y,X)
```

se obţin rezultatele:

```
K1 =
   1   2   2   4
   3   4   6   8
   3   6   4   8
   9  12  12  16
   5  10   6  12
  15  20  18  24

K2 =
   1   2   2   4
   3   4   6   8
   5   6  10  12
   3   6   4   8
   9  12  12  16
  15  18  20  24
```

10.8. Pătratul magic

Pătratul *magic* de ordinul n este o matrice n x n, construită cu întregii de la 1 la n^2, care are suma elementelor de pe fiecare linie, coloană, diagonală sau anti-diagonală principală, egală. Pătratul *magic* are suma scalată dublu stochastic. Funcţia *magic* se apelează cu sintaxa:

A=*magic*(n)

unde argumentul n trebuie să fie mai mare sau egal cu 3.

Exemplul 10.8.1. Să se genereze pătratul *magic* de ordinul 4.
Cu secvenţa MATLAB:

```
M = magic(4)
```

se obţine rezultatul:

```
M =
   16   2   3  13
    5  11  10   8
    9   7   6  12
    4  14  15   1
```

care este o matrice scalată dublu stochastic şi care are suma elementelor pe fiecare coloană sau linie, cât şi pe diagonală, egală cu 34.

10.9. Matricea Pascal

MATLAB poate genera o matrice pătrată având elementele triunghiului lui Pascal. Triunghiul lui Pascal conţine coeficienţii binomiali ai decompunerii $(a+b)^n$:

$$(a+b)^n = a^n + C_n^1 a^{n-1} b + C_n^2 a^{n-2} b^2 + \ldots + C_n^{n-1} a b^{n-1} + b^n$$

pentru orice număr natural n. De exemplu, triunghiul lui Pascal cu coeficienţii decompunerii unui binom de ordinul patru are următoarea formă:

```
0          1
1        1   1
2      1   2   1
3    1   3   3   1
4  1   4   6   4   1
```

Dacă numerotarea liniilor triunghiului lui Pascal se face începând cu zero, aşa cum este prezentat mai sus, fiecare linie din triunghi va conţine coeficienţii descompunerii binomului $(a+b)^n$, unde n este numărul liniei. Astfel, linia cu numărul trei conţine coeficienţii descompunerii binomului $(a+b)^3$, care sunt: 1, 3, 3, 1.

Funcţia *pascal* se apelează cu sintaxa:

P=*pascal*(n)

Astfel, instrucţiunea:

```
P=pascal(5)
```

returnează matricea Pascal de ordinul 5:

```
P =
    1   1   1   1   1
    1   2   3   4   5
    1   3   6  10  15
    1   4  10  20  35
    1   5  15  35  70
```

la care elementele din partea stângă a antidiagonalei principale sunt elementele triunghiului lui Pascal. De multe ori însă, apelarea funcţiei *pascal* se face cu una dintre sintaxele:

P=*abs*(*pascal*(n,1)) P=*abs*(*pascal*(n,2))

Pentru n=5, instrucţiunile:

```
P1=abs(pascal(5,1))
P2=abs(pascal(5,2))
```

returnează rezultatele:

```
P1 =                    P2 =
    1   0   0   0   0       0   0   0   0   1
    1   1   0   0   0       0   0   0   1   4
    1   2   1   0   0       0   0   1   3   6
    1   3   3   1   0       0   1   2   3   4
    1   4   6   4   1       1   1   1   1   1
```

Acestea conţin elementele triunghiului lui Pascal ordonate astfel: matricea P1 conţine elemetele triunghiului lui Pascal ordonate pe linie, în partea triunghiulară din colţul stâng, în timp ce matricea P2 conţine elemetele triunghiului lui Pascal ordonate pe coloană, în partea triunghiulară din colţul drept. De exemplu, pentru extragerea coeficienţilor descompunerii binomiale $(a+b)^8$, se foloseşte secvenţa:

```
n=8
P=abs(pascal(n+1,1));
Coef=P(n+1,1:n+1)
```

obţinându-se rezultatul:

```
Coef = [ 1   8   28   56   70   56   28   8   1 ]
```

10.10. Matricea Toeplitz

Matricea Toeplitz este definită prin vectorii elementelor primei coloane şi ai primei linii. O matrice Toeplitz simetrică este definită doar de un singur vector. Funcţia *toeplitz* se apelează cu una dintre sintaxele:

T=*toeplitz*(c) T=*toeplitz*(c,l)

Dacă funcţia *toeplitz* este apelată cu un singur vector argument, matricea returnată este o matrice Toeplitz pătrată, ale cărei primă coloană şi primă linie sunt elementele vectorului argument, iar pe diagonala principală şi pe toate diagonalele paralele cu aceasta se găsesc elementele definite în linia sau coloana amintită. În schimb, dacă funcţia *toeplitz* este apelată cu doi vectori argument de dimensiuni m şi n, matricea returnată are dimensiunea m x n, a cărei primă coloană este definită de primul vector argument, iar prima linie de elementele celui de-al doilea vector argument. Dacă primul element al primului vector (prima coloană) diferă de primul element al celui de-al doilea vector (prima linie), elementul coloanei are prioritate faţă de elementul liniei.

Exemplul 10.10.1. Să se genereze o matrice Toeplitz hermitică, definită de vectorul c = [1 2 3 4 5]..
Cu secvenţa MATLAB:

```
c = [1 2 3 4 5];
T1 = toeplitz(c)
```

se obţine rezultatul:

```
T1 =
     1     2     3     4     5
     2     1     2     3     4
     3     2     1     2     3
     4     3     2     1     2
     5     4     3     2     1
```

Exemplul 10.10.2. Să se genereze o matrice Toeplitz, definită de vectorii: c=[10 5 6] (prima coloană) şi l= [1 2 3 4] (prima linie).
Cu secvenţa MATLAB:

```
c = [10 5 6 ];      l= [1 2 3 4];
T = toeplitz(c,l)
```

se obţine rezultatul:

```
T =
    10     2     3     4
     5    10     2     3
     6     5    10     2
```

Se observă prioritatea elementului vectorului care defineşte prima coloană (primul vector argument, c) faţă de elementul vectorului care defineşte prima linie (al doilea vector argument, l)

10.11. Matricea Vandermonde

Matricea Vandermonde este definită de expresia:

$$V(i,j) = c(i)^{(n-j)}$$

unde c este vectorul cu n elemente, asociat penultimei coloane.
De exemplu:

$$V = \begin{bmatrix} 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \\ 3^3 & 3^2 & 3 & 1 \\ 4^3 & 4^2 & 4 & 1 \end{bmatrix}$$

este o matrice Vandermonde, definită de vectorul c=[1 2 3 4].
Funcţia *vander* se apelează cu sintaxa:
V= *vander*(c)

Exemplul 10.11.1. Să se genereze o matrice Vandermonde definită prin vectorul c = [2 3 4 5].
Cu secvenţa MATLAB:

```
c = [ 2 3 4 5 ];
V = vander(c)
```

se obţine rezultatul:


```
V =
     8     4     2     1
    27     9     3     1
    64    16     4     1
   125    25     5     1
```

10.12. Matricea Wilkinson

Matricea Wilkinson de ordinul n are elementele definite prin relaţia:

$$W(i,j) = \begin{cases} \dfrac{|n-2i+1|}{2} & , i=j \\ 1 & , i=j+1 \\ 0 & , \text{în rest} \end{cases}$$

pentru i∈[1, n]; de exemplu:

$$W = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

este o matrice Wilkinson de ordinul 5.
Funcţia *wilkinson* se apelează cu sintaxa:
W=*wilkinson*(n)

Exemplul 10.12.1. Să se genereze o matrice Wilkinson de ordinul 4. Cu secvenţa:

```
W=wilkinson(4)
```

se obţine rezultatul:

```
W =
    1.5000    1.0000         0         0
    1.0000    0.5000    1.0000         0
         0    1.0000    0.5000    1.0000
         0         0    1.0000    1.5000
```

11. CALCULE CU MATRICE

În multe domenii formularea matriceală a problemelor a condus la simplificarea metodelor de rezolvare şi a făcut posibilă extinderea unor soluţii deja cunoscute la domenii noi.

11.1. Manipularea matricelor

MATLAB-ul oferă următoarele funcţii pentru manipularea matricelor:

diag	Creează sau extrage diagonalele matricei;
fliplr	Roteşte matricea în jurul axei verticale (inversează ordinea coloanelor);
flipud	Roteşte matricea în jurul axei orizontale (inversează ordinea liniilor);
reshape	Schimbă dimensiunile matricei (redimensionează matricea);
rot90	Roteşte matricea cu un multiplu de 90 de grade;
tril	Extrage matricea inferior triunghiulară dintr-o matrice;
triu	Extrage matricea superior triunghiulară dintr-o matrice;
:	Specifică indicii, rearanjează sau decupează o submatrice.

11.1.1. Extragerea submatricelor prin indici

Elementele individuale ale unei matrice se apelează cu numele acesteia, urmat de doi indici, cuprinşi între paranteze rotunde şi separaţi de virgulă. Primul indice semnifică linia, iar al doilea coloana în care se găseşte elementul apelat.
De exemplu instrucţiunea:

A(3,3) = A(1,3) + A(3,1)

are ca rezultat înlocuirea elementului din linia 3 şi coloana 3, A(3,3), al matricei A, cu suma dintre elementul liniei 1 - coloanei 3, A(1,3), şi elementul din linia 3 - coloana 1.

Indicii pot fi scalari sau vectori; indicii vectori permit definirea unor submatrice, procedeu prin care se pot referi părţi disparate dintr-o matrice. Spre exemplu, instrucţiunea:

B=M(1:5,3)

extrage submatricea B de dimensiunea 5x1, constând din primele 5 elemente din coloana a 3-a a matricei M. De asemenea, instrucţiunea:

C=M(1:5,7:10)

extrage submatricea C cu dimensiunea 5x4, constituită din elementele primelor 5 linii (liniile 1-5) şi ale ultimelor 4 coloane (coloanele 7-10) ale matricei M.

Utilizarea numai a semnului două puncte „:" în locul indicilor pentru linii sau pentru coloane, presupune considerarea tuturor elementelor pe linii sau respectiv pe coloane. Astfel, instrucţiunea:

D=M(:,3)

extrage din matricea M, o submatrice D care conţine toate liniile (datorită semnului două puncte) din coloana a 3-a, în timp ce instrucţiunea:.

E=M(1:5,:)

extrage submatricea E care conţine liniile de la 1 la 5 ale tuturor coloanelor matricei M.

Crearea matricelor mari, precum şi manipularea acestora se face cu multă

flexibilitate dacă se utilizează indici vectori.
Spre exemplu, instrucţiunea:

A(:,[3,5,10]) = B(:,1:3)

înlocuieşte coloanele 3, 5 şi 10 ale matricei A cu primele 3 coloane ale matricei B. În general, dacă indicii x şi y sunt vectori cu componente întregi, atunci instrucţiunea:

B=A(x,y)

creează o submatrice B, care conţine elementele din liniile x ale coloanelor y ale matricei A.

Inversarea coloanelor unei matrice A se face cu instrucţiunea :

A=A(:,n:-1:1)

iar inversarea liniilor se face cu instrucţiunea :

A=A(n:-1:1,:)

De o mare importanţă este utilizarea fără indici, care are ca efect transformarea matricei într-un vector coloană, citind coloanele una după alta.
Astfel:

```
A = [ 1 2; 3 4 ]
b = A(:)
```

are ca rezultat transformarea unei matrice A într-un vector coloană:

```
A =                b =
   1   2                1
   3   4                3
                        2
                        4
```

Dacă în partea stângă a expresiei este asignată o instrucţiune de forma A(:), matricea A trebuie deja să existe dintr-o utilizare anterioară. Spre exemplu, secvenţa:

```
A = [ 1 2; 3 4 ]
A(:) = 11:14
```

returnează matricea:

```
A =
   11   13
   12   14
```

Pentru extragerea vectorilor cu elemente decupate din alţi vectori, se folosesc formele:

j:k	- selectează elementele [j, j+1, ..., k] ale unui vector; dacă j>k, vectorul rezultat este gol.
j:i:k	- selectează elementele [j, j+i, j+2i, ...,k] ale unui vector ; vectorul rezultat este gol dacă i>0 şi j>k sau dacă i<0 şi j<k.

În cazul selectării liniilor sau coloanelor matricelor, se folosesc formele:

A(:,j)	- selectează coloana j a matricei A;
A(i,:)	- selectează linia i a matricei A;
A(:,:)	- selectează întreaga matrice A;
A(j:k)	- selectează elementele A(j), A(j+1),... , A(k) ale vectorului A;
A(:,j:k)	- selectează toate liniile şi coloanele de la j la k, A(:,j),A(:,j+1),...A(:,k), ale matricei A;
A(:)	- selectează toate elementele matricei A, privite ca o singură coloană (începând cu prima).

Exemplul 11.1.1. Să se selecteze elementele din poziţiile 3-5 şi 2,4,6 ale vectorului: A=[2 4 7 3 5 1 0].
Cu secvenţa MATLAB:

```
A=[2 4 7 3 5 1 0];
B=A(3:5)
C=A(2:2:6)
```

se obţine rezultatul:

B = [7 3 5]
C = [4 3 1]

Exemplul 11.1.2. Generaţi un vector C prin întreţeserea elementelor vectorilor: A=[1 3 5 7] şi B=[2 4 6 8].
Cu secvenţa MATLAB:

```
A=[1 3 5 7]; B=[2 4 6 8];
n=length(A)+length(B);
C(1:2:n)=A; C(2:2:n)=B
```

se obţine rezultatul:

C = [1 2 3 4 5 6 7 8]

Exemplul 11.1.3. Fie $A=\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \end{bmatrix}$.

Să se scrie instrucţiunile care permit selectarea din matricea A a:
- liniei a 2-a ;
- coloanei a 3-a;
- submatricei formată din liniile 1-2 şi coloanele 4-6;
- submatricei formată din liniile 1, 4 şi coloanele 2, 4,5 ;

Cu secvenţa MATLAB:

```
A=[1 2 3 4 5 6; 2 3 4 5 6 1; 3 4 5 6 1 2;...
   4 5 6 1 2 3; 5 6 1 2 3 4; 6 1 2 3 4 5]
B=A(2,:)
C=A(:,3)
D=A(1:2,4:6)
E=A([1,4],[2,4:5])
```

se obţin rezultatele:

B = [2 3 4 5 6 1]

C =
3
4
5
6
1
2

D =
4 5 6
5 6 1

E =
2 4 5
5 1 2

11.1.2. Extragerea submatricelor prin vectori cu elemente 0 - 1

Vectorii cu elemente 0 şi 1, creaţi de operatorii logici sau relaţionali, au proprietăţi deosebite, care pot conduce la scrierea unor programe foarte compacte.

Dacă A este o matrice m x n şi L un vector de lungime m, cu elemente 0 şi 1, instrucţiunea:

B=A(L,:)

returnează în matricea B toate elementele din liniile matricei A pentru care elementul corespunzător ca poziţie din vectorul L este 1. Prin acest procedeu se extrage o submatrice cu un vector returnat de funcţiile logice. De exemplu, dacă:

L=[0 1 0 1 1]

instrucţiunea B=A(L,:) elimină liniile 1 şi 3 ale matricei A.

Pentru eliminarea coloanelor expresia de mai sus se transformă în:

C=A(:,L)

Numărul de elemente ale lui L trebuie să fie egale cu numărul de coloane ale matricei A.

Exemplul 11.1.4. Să se extragă dintr-o matrice X toate liniile care au elementul din coloana a 3-a mai mare sau egal cu 1.
Cu secvenţa MATLAB:

```
X=[ 0 1  1
    7 0 -5
    0 5  8 ];
L = X(:,3)>=1;
Y = X(L,:)
```

se obţine rezultatul:

Y =
0 1 1
0 5 8

11.1.3. Asamblarea matricelor mari

O matrice poate fi definită pe baza unor alte matrice de dimensiuni inferioare, ca în exemplul:

C = [A eye(4); ones(A) A^2]

Matricea C este compusă din matricea A de dimensiunea 4x4, matricea identitate 4x4, matricea unitate de dimensiunile matricei A şi matricea A ridicată la pătrat. Matricele folosite în asamblare trebuie să fie consistente ca dimensiuni, adică să determine blocuri care se integrează compact în tabloul matricei rezultat; în caz contrar se obţin mesaje de eroare.

11.1.4. Redimensionarea unei matrice

Funcţia *reshape* redimensionează o matrice A cu dimensiunea l x p într-o altă matrice B, cu dimensiunea m x n. Pentru ca operaţia de redimensionare să fie posibilă este necesar ca matricea A să aibă acelaşi număr de elemente cu matricea B (l x p = m x n).

Se apelează cu sintaxa:

B=*reshape*(A,m,n)

Elementele matricei B - în ordinea succesivă a coloanelor şi pe fiecare coloană de sus în jos - sunt elementele matricei argument A (citite de sus în jos şi de la stânga la dreapta).

Exemplul 11.1.5. Să se redimensioneze matricea A cu dimensiunea 3x4 în matricea B cu dimensiunea 2x6.

$$A=\begin{bmatrix}1 & 4 & 7 & 10\\ 2 & 5 & 8 & 11\\ 3 & 6 & 9 & 12\end{bmatrix}$$

Cu secvenţa MATLAB:

```
A=[1 4 7 10;2 5 8 11;3 6 9 12];
B = reshape(A,2,6)
```

se obţine rezultatul:

```
B =
   1   3   5   7   9  11
   2   4   6   8  10  12
```

11.1.5. Rotirea matricei în jurul unei coloane sau linii

Funcţiile *fliplr* şi *flipud* inversează ordinea liniilor sau coloanelor matricelor. Practic, ele realizează o rotire a matricei în jurul unei coloane sau a unei linii. Se apelează cu sintaxele:

Y=fliplr(X) *Z=flipud*(X)

Funcţia *fliplr*(X) returnează matricea Y rezultată prin pivotarea matricei argument în jurul ultimei coloane, operaţie prin care aceasta devine prima coloană.

Funcţia *flipud(X)* returnează matricea Z rezultată prin pivotarea matricei argument în jurul ultimei linii, operaţie prin care aceasta devine prima linie.

Exemplul 11.1.6. Să se rotească matricea $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$

- în jurul unei coloane;
- în jurul unei linii.

Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 9];
B=fliplr(A)
C=flipud(A)
```

se obţin rezultatele:

```
B =                    C =
   3   2   1              7   8   9
   6   5   4              4   5   6
   9   8   7              1   2   3
```


11.1.6. Rotirea matricei cu multipli de 90 de grade în jurul unui element

Rotirea unei matrice cu multipli de 90 de grade, în sens trigonometric (contrar sensului orar) se face cu funcţia *rot90*; se apelează cu una dintre sintaxele:

B=*rot*90(A) B=*rot*90(A,k)

Argumentul opţional k este multiplul de 90 grade cu care se roteşte matricea, în sens trigonometric, dacă este pozitiv şi în sens contrar, dacă este negativ.

Exemplul 11.1.7. Să se rotească în jurul unui element, matricea $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$

- cu 90 de grade în sens trigonometric;
- cu 180 de grade în sens orar.

Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 9];
B=rot90(A,1)
C=rot90(A,-2)
```

se obţin rezultatele:

```
B =                    C =
   3   6   9              9   8   7
   2   5   8              6   5   4
   1   4   7              3   2   1
```

11.1.7. Matrice diagonale şi triunghiulare

Crearea unei matrice diagonale se face cu funcţia *diag*; se apelează cu una dintre sintaxele:

Y=*diag*(X) Y=*diag*(X,k)

X este vectorul sau matricea asupra căreia se operează, iar argumentul opţional k indică diagonala acesteia, cu următoarea semnificaţie:

k=0, diagonala principală;
k>0, indică diagonala k de deasupra celei principale;
k<0, indică diagonala k de sub cea principală.

Crearea unei matrice superior şi inferior triunghiulare, foloseşte funcţiile MATLAB *tril* şi *triu*, apelate cu una dintre sintaxele:

Y=*tril*(X) Y=*tril*(X,k)
Y=*triu*(X) Y=*triu*(X,k)

unde argumentele au aceeaşi semnificaţie ca mai sus.

Dacă X este un vector cu n componente, funcţia *diag*(X,k) generează o matrice pătrată de ordinul n+abs(k), cu elementele lui X pe diagonala k. Instrucţiunea *diag*(X) generează o matrice cu elementele lui X pe diagonala principală.

Dacă X este o matrice, funcţia *diag*(X,k) extrage un vector coloană format din elementele diagonalei k a matricei X. Apelată fără argumentul k, *diag*(X) extrage elementele diagonalei principale din matricea X.

Funcţia *triu*(X) extrage matricea superior triunghiulară din matricea X, în timp ce *triu*(X,k) înlocuieşte cu zero toate elementele matricei X de sub diagonala k.

Funcţia *tril*(X) extrage matricea inferior triunghiulară din matricea X, în timp ce *tril*(X,k) înlocuieşte cu zero toate elementele matricei X de deasupra diagonalei k.

Exemplul 11.1.8. Să se aplice funcţiile *diag*, *tril* şi *triu* matricei $X=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$.

Cu secvenţa de instrucţiuni:

```
X=[1 2 3;4 5 6;7 8 9];
A = diag(diag(X))
B = triu(X)
C = tril(X)
A1=diag(X)
B1=triu(X,1)
C1=tril(X,1)
```

se obţin rezultatele:

```
A =                B =                C =
   1   0   0          1   2   3          1   0   0
   0   5   0          0   5   6          4   5   0
   0   0   9          0   0   9          7   8   9

A1 =               B1 =               C1 =
   1                  0   2   3          1   2   0
   5                  0   0   6          4   5   6
   9                  0   0   0          7   8   9
```

11.2. Analiză matriceală

Acest subcapitol face referire la funcţiile MATLAB:

det	Calculează determinantul unei matrice;
inv	Calculează inversa unei matrice;
rank	Calculează rangul matricei;
trace	Calculează urma unei matrice;
cond	Estimează numărul de condiţionare al matricei;
condest	Estimează numărul de condiţionare al matricei;
norm	Calculează norma vectorului sau matricei;
rcond	Estimează numărul de condiţionare al matricei;

11.2.1. Determinantul unei matrice

Calculul determinantului unei matrice se face cu funcţia *det*; se apelează cu sintaxa:

D = *det*(X)

Exemplul 11.2.1. Calculaţi determinanţii următoarelor matrice:

$$A=\begin{bmatrix}1&2\\3&4\end{bmatrix}\qquad B=\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

Cu secvenţa MATLAB:

```
A=[1 2;3 4];B=[1 2 3;4 5 6;7 8 9];
A1=det(A)
B1=det(B)
```

se obţin rezultatele:

A1=-2 B1=0

11.2.2. Inversa unei matrice

Prin definiţie, inversa unei matrice pătrate A, este matricea A^{-1}, care satisface relaţia:

$$A\cdot A^{-1}=A^{-1}\cdot A=I$$

unde I este matricea unitate. O matrice este inversabilă numai dacă determinantul acesteia este diferit de zero, adică dacă este nesingulară.

Inversa unei matrice se calculează cu funcţia *inv*; se apelează cu sintaxa:

Y = *inv*(X)

Funcţia se utilizează frecvent pentru rezolvarea sistemelor de ecuaţii liniare Ax=b. Soluţia sistemului este x=inv(A)*b.

Exemplul 11.2.2. Calculaţi inversa matricei: $A=\begin{bmatrix}2&1\\-1&1\end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[2 1; -1 1];
B=inv(A)
```

se obţine rezultatul:

```
B =
   0.3333  -0.3333
   0.3333   0.6667
```

11.2.3. Rangul unei matrice

O matrice X nenulă cu m linii şi n coloane are rangul k, dacă X are un minor nenul de ordin k, iar toţi minorii de ordin mai mare decât k (dacă există) sunt nuli. Rangul unei matrice este un scalar $1 \le k \le \min(m,n)$.

Rangul unei matrice reprezintă numărul de linii sau coloane liniar independente ale acesteia şi se determină cu funcţia *rank*; se apelează cu una dintre sintaxele:

r= *rank*(X) r= *rank*(X,tol)

Apelată cu sintaxa r=*rank*(X,tol), funcţia *rank* returnează numărul de valori singulare ale lui X, mai mari decât parametrul opţional tol.

Exemplul 11.2.3. Să se determine rangul matricei $A = \begin{bmatrix} 3 & 2 & -5 & 4 \\ 3 & -1 & 3 & -3 \\ 3 & 5 & -13 & 11 \end{bmatrix}$.

Deoarece toţi minorii de ordinul al treilea ai matricei A sunt nuli:

$$\begin{vmatrix} 3 & 2 & -5 \\ 3 & -1 & 3 \\ 3 & 5 & -13 \end{vmatrix} = \begin{vmatrix} 3 & -5 & 4 \\ 3 & 3 & -3 \\ 3 & -13 & 11 \end{vmatrix} = \begin{vmatrix} 3 & 2 & 4 \\ 3 & -1 & -3 \\ 3 & 5 & 11 \end{vmatrix} = \begin{vmatrix} 2 & -5 & 4 \\ -1 & 3 & -3 \\ 5 & -13 & 11 \end{vmatrix} = 0$$

şi deoarece există minori de ordinul al doilea nenuli, de exemplu:

$$\begin{vmatrix} 3 & 2 \\ 3 & -1 \end{vmatrix}$$

rezultă că rangul matricei A este:

rang A = 2

Acest rezultat se obţine cu secvenţa MATLAB:

```
A = [3 2 -5 4;3 -1 3 -3;3 5 -13 11];
k = rank(A)
```

care returnează: k = 2

11.2.4. Urma unei matrice

Urma matricei - suma elementelor de pe diagonala principală - se calculează cu funcţia *trace*; se apelează cu sintaxa:

u=*trace*(A)

Exemplul 11.2.4. Să se calculeze urma matricei $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[1 2 3;4  5 6;7 8 0];
B=trace(A)
```

se obţine rezultatul:

B = 6

11.2.5. Calculul normelor vectorilor şi matricelor

Norma este un scalar care dă o măsură a „mărimii" elementelor unei matrice sau vector. Normele vectorilor şi matricelor se calculează cu funcţia *norm*; se apelează cu una dintre sintaxele:

n=*norm*(X) n=*norm*(X,p) n=*norm*(X,'fro')

- Pentru matrice sunt definite următoarele tipuri de norme:

norm(X) sau *norm*(X,2) - returnează norma 2 (cea mai mare valoare singulară a lui X):

$$\|X\|_2 = \sqrt{\rho(X^* X)} = \max_{\lambda \in \sigma(X)} |\lambda|$$

unde mulţimea valorilor proprii ale matricei X - spectrul acesteia - s-a notat cu σ(X), iar ρ(X) şi λ sunt raza spectrală şi valorile singulare ale matricei X.

norm(X,1) - returnează norma 1 a lui X (cea mai mare sumă a elementelor de pe coloană):

$$\|X\|_1 = \max_{1 \le j \le N} \left(\sum_{k=1}^{N} |x_{jk}| \right)$$

norm(X,inf) - returnează norma infinită a lui X (cea mai mare sumă a elementelor de pe linie):

$$\|X\|_\infty = \max_{1 \le k \le N} \left(\sum_{j=1}^{N} |x_{jk}| \right)$$

norm(X,'fro') - returnează norma-F (Frobenius), calculată cu relaţia:

$$\|X\|_F = \sqrt{\sum_{j,k=1}^{N} |x_{jk}|^2} = \sqrt{|x_{11}|^2 + |x_{22}|^2 + \ldots + |x_{NN}|^2}$$

Pentru vectori sunt definite următoarele tipuri de norme:

norm(V,p);

$$\|V\|_p = \sqrt[p]{\sum_{k=1}^{N} |V_k|^p} = \sqrt[p]{|V_1|^p + |V_2|^p + \ldots + |V_N|^p}$$

norm(V)=norm(V,2);

$$\|V\|_2 = \sqrt{\sum_{k=1}^{N} |V_k|^2} = \sqrt{|V_1|^2 + |V_2|^2 + \ldots + |V_N|^2}$$

norm(V)/*sqrt*(N) - returnează valoarea rădăcinii medii pătratice (RMS - valoarea efectivă);

$$\text{RMS} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} |V_k|^2} = \sqrt{\frac{|V_1|^2 + |V_2|^2 + \ldots + |V_N|^2}{N}}$$

norm(V,inf) - returnează valoarea maximă în modul;

$$\|V\|_\infty = \max |V_k|_{1 \le k \le N}$$

norm(V,-inf) - returnează valoarea minimă în modul;

$$\|V\|_{-\infty} = \min |V_k|_{1 \le k \le N}$$

11.2.6. Condiţionarea unei matrice

Importanţa cunoaşterii condiţionării unei matrice decurge din faptul că rezultatele obţinute prin calcul numeric corespund întotdeauna unei probleme perturbate, ca urmare a efectelor erorilor de rotunjire. Determinarea condiţionării are la bază teoria matematică a perturbaţiilor. Se spune că o funcţie f(x) este rău-condiţionată dacă pentru un x^* „apropiat" de x, funcţia $f(x^*)$ diferă „mult" de f(x). Invers, o funcţie se spune că este bine-condiţionată dacă pentru un x^* „apropiat" de x, funcţia $f(x^*)$ este „apropiată" de f(x). Trebuie remarcat că

termenii „apropiat" şi „mult" sunt vagi, motiv pentru care este necesară utilizarea numerelor de condiţionare.

Numărul de condiţionare al unei matrice indică sensibilitatea soluţiei unui sistem de ecuaţii liniare şi dă indicaţii despre precizia rezultatelor la inversarea matricelor şi la rezolvarea ecuaţiilor liniare. O matrice bine-condiţionată este o matrice relativ insensibilă la mici perturbaţii.

Numărul de condiţionare al unei matrice, calculat ca raportul dintre cea mai mare şi cea mai mică valoare singulară a acesteia, se determină cu funcţia *cond* ; se apelează cu sintaxa:

c=*cond*(X)

O funcţie mai performantă care permite calculul numărului de condiţionare este *rcond*; se apelează cu sintaxa:

c=*rcond*(X)

Dacă matricea X este bine-condiţionată, *rcond* are valoarea aproximativ 1.0, iar dacă e slab-condiţionată, *rcond* este aproximativ 0.0

Estimarea celui mai mic număr de condiţionare se face cu funcţia *condest*; se apelează cu sintaxa:

c=*condest*(X)

Exemplul 11.2.5. Calculaţi numerele de condiţionare pentru sistemul de ecuaţii:

$$\begin{cases} 6x_1 & +6.917x_2 & = 6.543 \\ x_1 & +1.152x_2 & = 1.095 \end{cases}$$

şi de asemenea pentru sistemul „perturbat":

$$\begin{cases} 6x_1 & +6.912x_2 & = 6.543 \\ x_1 & +1.152x_2 & = 1.095 \end{cases}$$

Cu secvenţa MATLAB:

```
A1=[6  6.917; 1 1.152];
A2=[6  6.912; 1  1.152];
c1=cond(A1); c2=cond(A2);
d1=condest(A1); d2=condest(A2);
r1=rcond(A1); r2=rcond(A2);
```

se obţin rezultatele:

```
c1 = 1.7234e+004          c2 =  Inf
d1 = 2.0845e+004          d2 =2.7129e+017
r1 = 5.4453e-005          r2 =4.1842e-018
```

de unde rezultă că sistemul este rău-condiţionat. Acest lucru se verifică şi prin rezolvarea celor două sisteme, cu valori „apropiate" ale coeficienţilor (diferenţă la a patra cifră).

Cu secvenţa MATLAB:

```
A1=[6  6.917; 1 1.152];
A2=[6  6.912; 1  1.152];
b=[6.543; 1.095]
x1=A1\b
x2=A2\b
```

se obţin rezultatele:

```
x1 = [ 7.3158     -5.4000 ]
```



```
x2 = [ -8.1041    7.0348 ] *1.0e+013
```

Slaba condiţionare a matricei sistemului se reflectă în diferenţele foarte mari ale soluţiilor.

11.3. Descompunerea şi factorizarea matricelor

Funcţiile MATLAB folosite pentru descompunerea şi factorizarea matricelor sunt:

chol	Calculează factorizarea Cholesky;
lu	Calculează descompunerea LU (lower-upper) a matricei;
qr	Efectuează descompunerea ortogonal-triunghiulară a matricei;
qz	Calculează descompunerea QZ a matricei;
pinv	Calculează pseudoinversa matricei;
rref	Reduce matricele la forma triunghiulară.

11.3.1. Valori şi vectori proprii

În determinarea soluţiilor nebanale ale ecuaţiei:

$$Ax = \lambda x$$

unde:

A - este o matrice patratică de ordinul n,
x - este vector coloană de ordinul n, iar
λ - este un scalar.

valorile x, respectiv λ, care satisfac ecuaţia de mai sus se numesc vectori proprii, respectiv valori proprii.

Funcţia *eig* calculează valorile şi vectorii proprii ai unei matrice pătrate; se apelează cu una dintre sintaxele:

V=*eig*(A) V=*eig*(A,B)
[V,D] = *eig*(A) [V,D] = *eig*(A,B) [V,D]=*eig*(A,'nobalance')

care oferă următoarele rezultate:

- V=*eig*(A) returnează un vector care conţine valorile proprii ale matricei A.
- [V,D]= *eig*(A) returnează o matrice diagonală D care conţine valorile proprii ale matricei A şi o matrice V ale cărei coloane sunt vectorii proprii corespunzători valorilor proprii, astfel încât:

  A * V = V * D
- [V,D]= *eig*(A,'nobalance') returnează valorile şi vectorii proprii fără a executa în prealabil o balansare. Dacă matricea conţine elemente mici, comparabile cu erorile de rotunjire, balansarea le scalează, făcându-le la fel de semnificative ca celelalte elemente ale matricei originale. Acest lucru conduce în final la vectori proprii incorecţi. În astfel de situaţii se utilizează opţiunea 'nobalance'.

  Dacă matricea are numai valori proprii de ordinul întâi (valorile proprii λ sunt distincte), atunci vectorii proprii sunt independenţi. Dacă vectorii proprii nu sunt independenţi, atunci matricea originală este neregulată. Chiar dacă o matrice este neregulată, soluţia funcţiei *eig* satisface ecuaţia:

  A * X = X * D

  Valorile şi vectorii proprii generalizaţi determină soluţiile nebanale ale ecuaţiei:

$$Ax = \lambda Bx$$

unde A şi B sunt matrice pătratice de ordinul n, iar λ este un scalar. Valorile lui λ care satisfac ecuaţia se numesc valori proprii generalizate şi valorile x corespunzătoare sunt vectorii proprii generalizaţi. Dacă B este o matrice nesingulară (det B $\neq$ 0, adică matricea este inversabilă), problema calculului valorilor şi vectorilor proprii se reduce la la o problemă standard de valori proprii prin înlocuirea lui A cu $B^{-1}A$, deoarece ecuaţia este echivalentă cu:

$$B^{-1}Ax = \lambda x$$

Calculul vectorilor şi valorilor proprii generalizate se face cu funcţia MATLAB *eig*; se apelează cu sintaxa:

V=*eig*(A,B) - returnează un vector care conţine valorile proprii generalizate, dacă A şi B sunt matrice pătratice,.

[V,D]=eig(A,B) - returnează o matrice diagonală D ce conţine valorile proprii generalizate şi o matrice V ale cărei coloane sunt vectorii proprii corespunzători, astfel încât:

A*V = B*V*D

Exemplul 11.3.1. Valorile şi vectorii proprii ai matricei $A = \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[-2 1 0; 1 -2 1; 0 1 -2];
[V,D]=eig(A)
```

se obţin rezultatele:

V =

0.5000	-0.7071	-0.5000
0.7071	0.0000	0.7071
0.5000	0.7071	-0.5000

D =

-0.5858	0	0
0	-2.0000	0
0	0	-3.4142

11.3.2. Calculul valorilor singulare

Funcţia *svd* returnează valorile singulare ale unei matrice. Calculul acestor valori este un mijloc sigur de determinare a rangului unei matrice de formă generală. Funcţia *svd* se apelează cu una dintre sintaxele:

d=*svd*(X) - returnează un vector care conţine valorile singulare;

[U,S,V] = *svd*(X) - returnează o matrice diagonală S cu aceleaşi dimensiuni ca X, având elementele diagonale nenegative (care sunt şi valorile singulare) în ordine descrescătoare şi matricele U şi V care satisfac relaţia:

X = U*S*V'

[U,S,V] = *svd*(X,0) - efectuează un calcul mai rapid al valorilor singulare.

Exemplul 11.3.2. Să se descompună în valori singulare matricea $A = \begin{bmatrix} 1 & 5 \\ 4 & 3 \end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[1 5;4 3];
d=svd(A)
[U,S,V]=svd(A)
```


se obţin rezultatele:

d =

954/143
3435/1348

U =

3177/4435	464/665
464/665	-3177/4435

S =

954/143	0
0	3435/1348

V =

521/991	-3725/4379
3725/4379	521/991

11.3.3. Factorizarea Cholesky

Factorizarea Cholesky este o metodă directă de rezolvare a unui sistem de ecuţii liniare de forma:

Ax=b

cu A matrice pozitiv definită.

Următoarele afirmaţii sunt echivalente:

1. matricea A este pozitiv definită;
2. $A=A^t$ şi det(A)>0 (A^t este notaţia matricei transpuse);
3. există o matrice unică R, superior triunghiulară, nesingulară, care satisface relaţia: $A=R^tR$.

Factorizarea Cholesky constă în descompunerea matricei coeficienţilor sistemului conform pct. 3, unde R este numit factorul Cholesky. Sistemul iniţial devine:

$R^tRx=b$

iar soluţia acestuia presupune rezolvarea succesivă a sistemelor: $R^ty=b$ şi apoi Rx=y.

Factorizarea Cholesky se calculează cu funcţia MATLAB *chol*; se apelează cu una dintre sintaxele:

R=*chol*(A) [R, p]=*chol*(X)

unde:

A - este o matrice pozitiv definită;
R - este o matrice superior triunghiulară, astfel încât R'*R=A;
p - este un scalar de test, egal cu zero dacă matricea A este pozitiv definită şi un întreg pozitiv în caz contrar.

Rezolvarea în MATLAB a unui sistem de ecuaţii liniare presupune etapele:

- se calculează factorul Cholesky, cu relaţia: R=chol(A);
- se rezolvă sistemul $R^ty=b$ cu relaţia : y=R'\b;
- se rezolvă sistemul Rx=y cu relaţia: x=R\y.

Aceste etape se pot scrie compact sub forma:

R=*chol*(A); x=R\R'\b

Exemplul 11.3.3. Calculaţi factorul Cholesky al matricei:

$$A = \begin{bmatrix} 9 & 1 & -1 & 0 \\ 1 & 9 & 0 & -1 \\ -1 & 0 & 9 & 1 \\ 0 & -1 & 1 & 9 \end{bmatrix}$$

Cu secvenţa MATLAB:

```
A=[9,1,-1,0;1,9,0,-1;-1,0,9,1;0,-1,1,9];
[R,p]=chol(A)
```

se obţine rezultatul:

```
R =
   3.0000   0.3333  -0.3333        0
        0   2.9814   0.0373  -0.3354
        0        0   2.9812   0.3396
        0        0        0   2.9618
p =   0  (matrice pozitiv definită)
```

11.3.4. Factorizarea LU (lower-upper)

Prin factorizare LU, o matrice pătrată este descompusă sub forma produsului a două matrice triunghiulare: una inferior triunghiulară, cu elemente 1 pe diagonala principală (matricea L) şi cealaltă superior triunghiulară (matricea U). În MATLAB, factorizarea LU este utilizată pentru obţinerea inversei unei matrice cu funcţia *inv* şi pentru calculul determinantului cu funcţia *det*. Este totodată baza rezolvării ecuaţiilor liniare prin „împărţirea matricelor" obţinută cu operatorii \ şi /.

Factorizarea LU a unei matrice se face cu funcţia *lu*; se apelează cu una dintre sintaxele:

[L,U]=*lu*(X) - returnează o matrice superior triunghiulară U şi o matrice inferior triunghiulară permutată L, astfel încât X=L*U;

[L,U,P]=*lu*(X) - returnează o matrice superior triunghiulară în U, inferior triunghiulară în L şi permutarea matriceală în P, astfel încât L*U=P*X.

Exemplul 11.3.4. Să se factorizeze LU matricea $A=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 0];
[L,U]=lu(A)
```

se obţin rezultatele :

```
L =                                U =
   0.1429   1.0000        0            7.0000   8.0000        0
   0.5714   0.5000   1.0000                 0   0.8571   3.0000
   1.0000        0        0                 0        0   4.5000
```

L este o permutare a matricei inferior triunghiulare, iar U este o matrice superior triunghiulară.

Secvenţa MATLAB:

```
[L,U,P]=lu(A)
```

returnează următoarele rezultate:

```
L =                                U =                                P =
   1.0000   0        0                7.000   8.0000   0                 0   0   1
   0.1429   1.0000   0                0       0.8571   3.0000            1   0   0
   0.5714   0.5000   1.0000           0       0        4.5000            0   1   0
```


Inversa unei matrice, X=inv(A), poate fi calculată şi cu expresia:

X=*inv*(U)**inv*(L)

iar determinantul:

D=*det*(A)

poate fi calculat cu expresia:

D=*det*(L)**det*(U)

Rezolvarea sistemelor de ecuaţii Ax=b prin factorizarea LU presupune soluţionarea succesivă a sistemelor:

y=L\b şi x=U\y

11.3.5. Factorizarea QR

Factorizarea QR este o descompunere a unei matrice ca produs al unei matrice ortonormale, Q, cu o matrice superior triunghiulară, R.

Factorizarea „QR" se utilizează pentru rezolvarea sistemelor de ecuaţii liniare cu mai multe ecuaţii decât necunoscute. Cea mai bună soluţie a unor astfel de sisteme, în sensul celor mai mici pătrate, este calculată cu instrucţiunea: x=A\b, care utilizează această factorizare. Prin utilizarea factorizării QR, soluţia x este calculată în doi paşi:

y=Q'*b; x=R\y

Descompunerea ortogonal-triunghiulară (factorizarea QR) se realizează cu funcţia *qr*; se apelează cu una dintre sintaxele:

[Q,R]=*qr*(X) - returnează matricea triunghiular superioară R de aceeaşi dimensiune cu X şi matricea Q, asfel încât X=Q*R;

[Q,R,E]=*qr*(X) - returnează matricea permutată E a matricei superior triunghiulare R, cu elementele diagonalei descrescătoare şi matricea Q, asfel încât X*E=Q*R. triu(qr(X)) = R;

Exemple 11.3.5. Să se determine descompunerea QR a matricei $A=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$.

Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 0];
[Q,R]=qr(A)
```

se obţin rezultatele:

```
Q =                                R =
   -129/1048   379/419   881/2158      -1048/129  -5079/529  -1685/507
   -129/262    379/1257  -881/1079       0          379/419    1895/419
   -1040/1207  -379/1257  881/2158       0          0          -3722/1013
```

11.3.6. Pseudoinversa unei matrice

Pseudoinversa sau inversa generalizată Moore-Penrose a unei matrice se calculează cu funcţia MATLAB *pinv*; se apelează cu sintaxa:

X=*pinv*(A)

Pseudoinversa se utilizează, de regulă, pentru rezolvarea sistemelor de ecuaţii liniare, în sensul celor mai mici pătrate. Matricea X este de aceleaşi dimensiuni ca A'

(transpusa lui A) şi verifică relaţiile: A*X*A = A, X*A*X = X iar AX şi XA sunt matrice Hermitice.

Exemplul 11.3.6. Să se calculaeze pseudoinversa matricei $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}$.

şi să se verifice că AX şi XA sunt matrice Hermitice.
Cu secvenţa MATLAB:

```
A=[1 2 3;4 5 6;7 8 0];
X=pinv(A); B=A*X; C=X*A
```

se obţin rezultatele:

```
X =
  -1.7778   0.8889  -0.1111
   1.5556  -0.7778   0.2222
  -0.1111   0.2222  -0.1111

B =                                C =
   1.0000   0.0000   0.0000          1.0000   0.0000   0.0000
   0.0000   1.0000   0.0000          0.0000   1.0000   0.0000
   0.0000   0.0000   1.0000          0.0000   0.0000   1.0000
```


12. PRELUCRAREA DATELOR ŞI CALCULE STATISTICE

În acest capitol sunt prezentate următoarele funcţii MATLAB pentru prelucrarea datelor şi calcule statistice:

cumsum	Calculează suma cumulată;
cumprod	Calculează produsul cumulat;
corcoef	Calculează coeficienţii de corelaţie;
cov	Calculează matricea de covarianţă;
diff	Calculează diferenţele dintre numere succesive;
hist	Reprezintă histograma;
max	Determină cea mai mare componentă;
min	Determină cea mai mică componentă;
mean	Calculează valoarea medie;
median	Calculează valoarea mediană;
prod	Calculează produsul;
sort	Sortează elementele în ordine crescătoare;
std	Calculează abaterea standard;
sum	Calculează suma;
trapz	Calculează integrala folosind metoda trapezelor.

12.1. Maximul şi minimul

Funcţiile *max* şi *min* determină maximul şi minimul valorilor unui vector sau matrice; se apelează cu una dintre sintaxele:

M= *max*(X), m=*min*(X)

Dacă X este vector, aceste funcţii returnează un scalar egal cu cel mai mare (mic) element, iar dacă X este matrice, funcţiile returnează un vector linie care conţine elementele maxime (minime) din fiecare coloană.

Determinarea elementelor maxime (minime) ale unui vector, precum şi a indicelui (poziţiei) acestora, se face prin apelarea acestor funcţii cu una dintre sintaxele:

[M,I]=*max*(X) [m,I]=*min*(X)

În vectorul M, respectiv m, se memorează valorile maxime (minime), iar în vectorul I, indicii acestor valori. Dacă vectorul X are mai multe valori maxime (minime) identice, se reţine indicele primului element maxim (minim) găsit.

Pentru a crea o matrice C care conţine elementele maxime (minime) din două matrice A si B, de aceeaşi dimensiune, se foloseşte sintaxa:

C=*max*(A,B) C=*min*(A,B)

Exemplul 12.1.1. Să se determine valorile maxime şi minime ale unui vector şi ale unei matrice.
Cu secvenţa MATLAB:

```
V = [1 3 -9 0];
M =[1 2 3;-4 0 9;13 7 -10];
A= max(V)
```

```
B = min(V)
C = max(M)
D = min(M)
[Y,I] = max(M)
```

se obţin rezultatele:

A = 3 B = -9
C = [13 7 9] D =[-4 0 -10]
Y = [13 7 9] I =[3 3 2]

12.2. Media şi mediana

Media aritmetică a unui set de date se calculează cu relaţia:

$$\mu = \frac{\sum_{k=1}^{N} x_k}{N} = \frac{x_1 + x_2 + \ldots + x_N}{N}$$

Funcţia *mean* se apelează cu sintaxa:

m= *mean*(X)

Pentru vectori, funcţia *mean* returnează un scalar egal cu valoarea medie a elementelor, iar pentru matrice, returnează un vector linie conţinând valoarea medie a fiecărei coloane (media pe coloană).

Mediana este elementul de la mijlocul unui set de date ordonat crescător. Dacă numărul N al valorilor setului este impar, valoarea mediană este cea din poziţia (N+1)/2, iar dacă acesta este par, valoarea mediană este media elementelor din poziţiile (N/2) şi (N/2)+1, adică:

$$\text{median}(x) = \begin{cases} x\left(\frac{N+1}{2}\right), & \text{dacă } N = 2M+1 \\ \frac{x\left(\frac{N}{2}\right) + x\left(\frac{N}{2}+1\right)}{2}, & \text{dacă } N = 2M \end{cases}$$

Funcţia *median* se apelează cu sintaxa:

M=*median*(X)

Pentru vectori funcţia *median* returnează un scalar egal cu valoarea mediană a elementelor lui X, iar pentru matrice, returnează un vector linie conţinând valoarea mediană a fiecărei coloane (mediana pe coloană).

12.3. Sume şi produse

MATLAB are funcţii pentru calculul sumelor şi produselor coloanelor matricelor, precum şi funcţii pentru calculul sumelor şi produselor cumulate ale elementelor acestora.

Pentru calculul sumei elementelor unui vector:

$$s = \sum_{k=1}^{N} x_k$$

se foloseşte funcţia *sum* care se apelează cu sintaxa:

Y=*sum*(X)

Dacă argumentul X este un vector, funcţia *sum* returnează un scalar egal cu suma elementelor vectorului, iar dacă argumentul este o matrice, funcţia returnează un vector linie care conţine suma elementelor pe fiecare coloană.

Pentru calculul produsului elementelor unui vector:

$$p = \prod_{k=1}^{N} x_k$$

se foloseşte funcţia *prod* care se apelează cu sintaxa:

Y=*prod*(X)

Dacă argumentul X este un vector, funcţia *prod* returnează un scalar egal cu produsul elementelor vectorului, iar dacă argumentul este o matrice, funcţia returnează un vector linie care conţine produsul elementelor pe fiecare coloană.

Prin sumă cumulată a elementelor unui vector se înţelege vectorul ale cărui elemente sunt date de relaţia:

$$s_j = \sum_{k=1}^{j} x_k \ , \qquad j = 1, 2, \ldots, N$$

Se foloseşte funcţia *cumsum* care se apelează cu sintaxa:

Y=*cumsum*(X)

Dacă argumentul X este un vector, funcţia returnează un vector care conţine suma cumulată a elementelor vectorului, iar dacă argumentul este o matrice, funcţia returnează o matrice care pe fiecare coloană conţine suma cumulată a elementelor din coloana respectivă.

Prin produs cumulat al elementelor unui vector se înţelege vectorul ale cărui elemente sunt date de relaţia:

$$p_j = \prod_{k=1}^{j} x_k \ , \qquad j = 1, 2, \ldots, N$$

Se foloseşte funcţia *cumprod* care se apelează cu sintaxa:

Y=*cumprod*(X)

Dacă argumentul X este un vector, funcţia returnează un vector care conţine produsul cumulat al elementelor vectorului, iar dacă argumentul este o matrice, funcţia returnează o matrice care pe fiecare coloană conţine produsul cumulat al elementelor din coloana respectivă.

Exemplul 12.3.1. Să se calculeze suma, suma cumulată, produsul şi produsul cumulat al elementelor unui vector.

Cu secvenţa MATLAB:

```
V=[1 2 3 4 5];
S=sum(V)
SC=cumsum(V)
P=prod(V)
PC=cumprod(V)
```

se obţin rezultatele:

S = 15 SC = [1 3 6 10 15]
P = 120 PC = [1 2 6 24 120]

Exemplul 12.3.2. Să se calculeze suma, suma cumulată, produsul şi produsul cumulat al elementelor unei matrice.
Cu secvenţa MATLAB:

```
M=[1 2 3;4 5 6;7 8 9;4 2 1];
S=sum(M)
P=prod(M)
SC=cumsum(M)
PC=cumprod(M)
```

se obţin rezultatele:

```
S =[ 16   17   19]          P =[ 112  160  162]
SC =                        PC =
     1    2    3                 1    2    3
     5    7    9                 4   10   18
    12   15   18                28   80  162
    16   17   19               112  160  162
```

12.4. Sortarea elementelor unei matrice

Sortarea elementelor unui vector sau matrice în ordine crescătoare se face cu funcţia *sort* care se apelează cu una dintre sintaxele:

Y=*sort*(X) [Y,I] = *sort*(X)

Dacă argumentul X este matrice, funcţia sortează fiecare coloană a acesteia. A doua formă de apelare a funcţiei *sort* returnează în matricea Y elementele coloanelor sortate în ordine ascendentă şi în matricea I indicii elementelor sortate ale matricei X. Dacă elementele coloanei matricei se repetă (au valori egale), indicii acestora sunt returnaţi în ordinea în care au fost întâlniţi în matricea argument. Pentru matrice complexe, elementele sunt sortate după valoarea modulului.

Exemplul 12.4.1. Să se sorteze elementele unui vector, cu precizarea indicelui fiecărui element.
Cu secvenţa MATLAB:

```
V=[1 3 -5 0 0]
[Y,I]=sort(V)
```

se obţin rezultatele:

```
Y = [ -5   0   0   1   3]
I = [  3   4   5   1   2]
```

Exemplul 12.4.2. Să se sorteze elementele unei matrice, cu precizarea indicelui fiecărui element.
Cu secvenţa MATLAB:

```
X=[1 -2 3;5 -9 0;-10 3 0];
[Y,I] = sort(X)
```

se obţin rezultatele:

```
Y =                        I =
  -10  -9   0                 3   2   2
    1  -2   0                 1   1   3
    5   3   3                 2   3   1
```


Exemplul 12.4.3. Să se schimbe ordinea coloanelor matricei M, astfel încât elementele din prima linie să fie ordonate crescător. (Această ordonare este utilă la reprezentarea grafică a matricelor, dacă prima linie şi prima coloană reprezintă coordonatele axelor.)
Cu secvenţa MATLAB:

```
M=[7 2 -5 4 -1
   5 8 1 -6 -4
   2 0 -3 6 9];
M1 = (M(1,:))' ;
[M2,I] = sort(M1);
M3 = M(:,I)'
```

se obţine rezultatul:

```
M3 =
  -5  -1   2   4   7
   1  -4   8  -6   5
  -3   9   0   6   2
```

Exemplul 12.4.4. Să se schimbe ordinea liniilor matricei N, astfel încât elementele din coloana a doua să fie ordonate descrescător.
Cu secvenţa MATLAB:

```
N=[7 -2 4;-5 2 0;4 10 -4;-7 5 -1];
N1 = N(:,2);
[N2,I] = sort(N1);
I1 = flipud(I);
N3 = N(I1,:)
```

sau cu secvenţa scrisă compact:

```
N = [7  -2  4; -5  2  0; 4  10  -4; -7  5  -1];
[N2,I] = sort(N(:,2));
N3 = N(flipud(I),:)
```

se obţine rezultatul:

```
N3 =
   4  10  -4
  -7   5  -1
  -5   2   0
   7  -2   4
```

12.5. Varianţa şi dispersia datelor

Varianţa unui set de date se calculează cu relaţia:

$$\sigma^2 = \frac{\sum_{k=1}^{N}(x_k-\mu)^2}{N-1}$$

unde termenul $(x_k-\mu)$ reprezintă diferenţa sau abaterea lui x_k faţă de valoarea medie μ. Prin urmare, varianţa reprezintă pătratul abaterii medii pătratice a datelor faţă de medie. Abaterea medie pătratică (abaterea standard sau dispersia) este definită ca rădăcina

pătrată a varianţei, $\sigma=\sqrt{\sigma^2}$.

Funcţia MATLAB *std* calculează abaterea medie pătratică:

$$\sigma=\sqrt{\frac{\sum_{k=1}^{N}(x_k-\mu)^2}{N-1}}$$

Dacă X este un vector, funcţia *std* returnează un scalar care reprezintă abaterea medie pătratică a elementelor vectorului; pentru matrice, funcţia *std* returnează un vector linie conţinând abaterea medie pătratică a fiecărei coloane.

12.6. Coeficientul de corelaţie a datelor

Pentru a stabili dacă între două seturi de date, înregistrate în vectorii u şi v, există o dependenţă liniară, se calculează coeficientul de corelaţie:

$$R_{uv}=\frac{N\sum_{i=1}^{N}u_iv_i-\sum_{i=1}^{N}u_i\sum_{i=1}^{N}v_i}{\sqrt{\left[N\sum_{i=1}^{N}u_i^2-\left(\sum_{i=1}^{N}u_i\right)^2\right]\left[N\sum_{i=1}^{N}v_i^2-\left(\sum_{i=1}^{N}v_i\right)^2\right]}}$$

Funcţia MATLAB care calculează acest coeficient este *corrcoef*, se apelează cu una dintre sintaxele:

Rx=*corrcoef*(X)
Rxy=*corrcoef*(X,Y)

Dacă funcţia *corrcoef* are ca argument de intrare o singură matrice, se returnează o matrice care conţine pe poziţiile Rx(i,j) coeficienţii de corelaţie dintre coloana i şi coloana j. Matricea este simetrică, deoarece Rx(i,j)=Rx(j,i).

Apelată cu două argumente de intrare, X şi Y, matrice de aceeaşi dimensiune, funcţia returnează o matrice care conţine pe poziţiile Rxy(i,j) coeficienţii de corelaţie dintre fiecare coloană X şi fiecare coloană Y.

Exemplul 12.6.1. Fie următoarele funcţii: x=2+3t, y=3t şi z=2sin(t). Calculaţi coeficienţii de corelaţie Rxy şi Rxz, pentru domeniul t∈[0,10].
Cu secvenţa MATLAB:

```
t=0:10;
x=2+3*t; y=3*t; z=2*sin(t);
Rxy=corrcoef(x,y)
Rxz=corrcoef(x,z)
```

se obţin rezultatele:

```
Rxy =                    Rxz =
   1   1                      1.0000  -0.1167
   1   1                     -0.1167   1.0000
```

Se observă că între x şi y există o dependenţă liniară, deoarece Rxy(1,2) = Rxy(2,1) = 1. Deoarece Rxz(1,2) = Rxz(2,1) ≠ 1, x şi z nu sunt într-o relaţie liniară.

12.7. Valori nedefinite

Valoarea specială NaN corespunde în MATLAB lui Not-a-Number. În mod normal această valoare este rezultatul unor operaţii nedefinite, 0/0 sau ∞/∞, conform convenţiilor stabilite de standardul IEEE de aritmetică în virgula mobilă.

Pentru identificarea poziţiilor ocupate de valorile NaN într-un vector x, se utilizează instrucţiunea:

I = *find*(*isnan*(x));

care returnează în vectorul I indicii elementelor care sunt valori NaN. Cu instrucţiunea:

y = x(*find*(~*isnan*(x)));

se obţine un vector y care conţine toate elementele lui x ce nu sunt NaN (elimină valorile NaN dintr-un set de date). Practic, pentru a elimina valorile NaN dintr-un set de date, există următoarele două modalităţi:

x = x(~*isnan*(x)) sau x(*isnan*(x)) = [ ]

din care a doua formă se execută mai rapid.

Orice operaţie în care unul dintre operanzi este NaN returnează valoarea NaN, inclusiv pentru operaţiile dintre două valori NaN.

Exemplul 12.7.1. Să se elimine liniile unei matrice care au cel puţin o valoare nedeterminată (NaN).
Cu secvenţa MATLAB:

Y=X(*find*(~*any*(*isnan*(X)')),:)

aplicată matricei:

```
X=[ 1     2     5
    4    NaN   Inf/Inf
    2    0/0    14
    7     3      9]
```

se obţine rezultatul:

```
Y =
   1   2   5
   7   3   9
```

12.8. Eliminarea datelor eronate

Datele eronate sau cele care nu îndeplinesc o anumită condiţie impusă de utilizator pot fi înlocuite într-un mod asemănător cu valorile NaN.
Fie matricea A:

```
A =[9  1000  4  8
    1     6  5  5
    3     2  7  1];
```

Să se elimine coloanele matricei care conţin cel puţin un element ce se abate de la valoarea medie cu mai mult de 3 ori abaterea standard (legea celor trei sigma din Statistică).
Se foloseşte următoarea secvenţă MATLAB:

```
mv=mean(A);
sigma=std(A);
[n,p]=size(A);
e = ones(n,1);
dist=abs(A-e*mv);
mcond=dist<3*e*sigma;
Y=A(:,find(all(mcond)))
celim=p-length(find(all(mcond)))
```

Dacă variabila `celim` este egală cu zero, matricea Y va fi identică cu matricea A (nu se elimină nici o coloană). Dacă aceasta este diferită de zero, coloanele care nu îndeplinesc condiţia impusă sunt eliminate din matricea A.

12.9. Histograme

Histograma este un tip special de grafic, deosebit de relevant pentru prelucrări statistice. Funcţia MATLAB *hist*, pentru reprezentarea histogramei se apelează cu una dintre sintaxele:

hist(X) *hist*(X,N)
[N,Y]=*hist*(X) [N,Y]=*hist*(X,N)

unde X este setul de date analizat, iar N, numărul de intervale (implicit, N=10).

Dacă este apelată fără argumente de ieşire, funcţia realizează reprezentarea histogramei datelor din X. Apelată cu argumente de ieşire, returnează vectorii N şi Y ce pot fi utilizaţi ulterior pentru reprezentarea grafică a histogramei cu funcţia *bar*. Vectorul N conţine numărul de valori care se găsesc în fiecare interval corespondent din Y. Funcţia *bar* se apelează cu una dintre sintaxele:

bar(Y) *bar*(X,Y) [XX,YY]=*bar*(X,Y)

Apelată fără argumente de ieşire, funcţia *bar* reprezintă grafic histograma datelor din vectorul Y. Dacă se foloseşte şi argumentul de intrare X, acesta trebuie să fie ordonat crescător, pentru a fi posibilă reprezentarea grafică. Cu argumente de ieşire, funcţia returnează datele calculate în vectorii XX şi YY. Reprezentarea histogramei se face apoi cu funcţia *plot*:

plot(XX,YY)

13. MINIMIZAREA FUNCŢIILOR

Funcţiile folosite în problemele de minimizare sunt:

fmin Calculează minimul funcţiilor de o variabilă reală;
fmins Calculează minimul funcţiilor de mai multe variabile reale;
fzero Determină zeroul unei funcţii de o variabilă reală.

13.1. Calculul minimului funcţiilor de o variabilă

Funcţia *fmin* se utilizează pentru determinarea minimului unei funcţii de o singură variabilă; se apelează cu una dintre sintaxele:

x=fmin('F', x1, x2)
x=fmin('F', x1, x2,opţiuni)
[x, opţiuni]=*fmin*('F',x1,x2, opţiuni, p1, p2, ...)

şi returnează valoarea lui x pentru care funcţia F(x) este minimă în intervalul x1 < x < x2. 'F' este un şir care conţine numele unui fişier M în care se găseşte funcţia ce trebuie minimizată. Funcţia *fmin* acceptă până la 10 parametri adiţionali care pot fi transmişi funcţiei F(x).

Funcţia *fmin* poate fi apelată cu opţiuni. Pentru controlul opţiunilor se utilizează un vector. Numai 3 dintre cele 18 componente ale opţiunilor sunt utilizate de funcţia *fmin* (ceilalţi sunt lăsaţi pentru dezvoltări ulterioare ale MATLAB-ului):

1. dacă opţiune(1) este diferit de zero, sunt afişate valorile intermediare ale rezolvării. Implicit opţiune(1) este 0;
2. opţiune(2) impune toleranţa. Valoarea implicită este 1.e-4;
3. opţiune(14) controlează numărul maxim de paşi (valoarea implicită este 500).

Exemplul 13.1.1. Să se determine coordonatele minimului funcţiei sin(x) în intervalul [0, 2π]. Cu instrucţiunile:

```
xmin = fmin('sin', 0, 2*pi)
ymin = sin(xmin)
```

se obţine rezultatul:

xmin = 4.7124 ymin = -1.0000

Exemplul 13.1.2. Să se determine minimul funcţiei $f(x) = (x-3)^2 - 1$, pe intervalul (0, 5).

Se scrie un fişier funcţie; fie numele acestuia test.m, cu următorul conţinut:

```
function y = test(x)
y = (x-3).^2 -1;
```

Se apelează funcţia *fmin* cu instrucţiunile:

```
xmin = fmin('test',0 ,5)
ymin = test(xmin)
```

obţinându-se rezultatul:

xmin = 3 ymin = -1

13.2. Calculul minimului funcţiilor de mai multe variabile

Determinarea minimului unei funcţii de mai multe variabile cel mai apropiat de o estimare iniţială x0, impusă ca argument, se face cu funcţia *fmins*; se apelează cu una din *sintaxele*:

x=fmins('F', x0)
x=fmins('F', x0,opţiuni)
[x, opţiuni]=*fmins*('F',x0, opţiuni, [ ], p1, p2, ...)

'F' este numele unui fişier M care conţine funcţia obiectiv ce trebuie minimizată.

Funcţia *fmins* acceptă până la 10 parametri adiţionali p1, p2, ...

Această funcţie poate fi apelată cu opţiuni. Pentru controlul opţiunilor se utilizează un vector. Numai 3 dintre cele 18 componente ale opţiunilor sunt utilizate de funcţia *fmins* (ceilalţi sunt lăsaţi pentru dezvoltări ulterioare ale MATLAB-ului):

1. dacă opţiune(1) este diferit de zero, sunt afişate valorile intermediare ale calculului. Implicit opţiune(1) este 0;
2. opţiune(2) impune toleranţa. Valoarea implicită este 1.e-4;
3. opţiune(14) controlează numărul maxim de paşi (valoarea implicită este 500).

Pentru modificarea valorilor implicite ale opţiunilor la apelarea funcţiei *fmins*, trebuie precizate opţiunile (de la prima până la a optsprezecea). Pentru detalii privind semnificaţia argumentului opţiuni, utilizatorul poate folosi comanda MATLAB:

```
help foptions
```

În cazul funcţiilor cu mai multe minime locale este importantă alegerea valorii iniţiale x0.

Exemplul 13.2.1. Să se determine minimul funcţiei:

$$F(x_1,x_2)=100(x_2-x_1^2)^2+(1-x_1)^2+2$$

cel mai apropiat de $(x_1,x_2)=(-1,1)$.

Se scrie fişierul funcţie cu conţinutul:

```
function F=test(x)
F= 100*(x(2)-x(1)^2)^2+(1-x(1))^2+2;
```

şi se înregistrează cu numele **test1.m**.

Cu secvenţa:

```
[xmin, optiuni]=fmins('test1',[-1,1]);
xmin
Fmin=test1(xmin)
Nr=optiuni(10)
```

se obţin rezultatele:

xmin = [1.0000 1.0000]
Fmin= 2.0000
Nr = 175

Nr=175 este numărul paşilor de calcul.

Exemplul 13.2.2. Fie funcţia:

$$F(x_1,x_2,a)=100(x_2-x_1^2)^2+(a-x_1)^2+2$$

unde a este un parametru. Să se determine minimul funcţiei $F(x_1,x_2,a)$, pentru estimarea iniţială: x0=(-1.5, .5) şi a=2.

Se scrie fişierul funcţie cu conţinutul:

```
function F=test2(x,a)
F= 100*(x(2)-x(1)^2)^2+(a-x(1))^2+2;
```

şi se înregistrează cu numele **test.m**.

Cu secvenţa:

```
x=[-1.5,.5]; a=3;
[xmin,opt]=fmins('test2',x,[0,1.e-8],[],a);
xmin
Fmin=test2(xmin,a)
Nr=opt(10)
```

se obţin rezultatele:

xmin = [3.0000 9.0000]
Fmin= 2.0000
Nr = 248

care arată că valoarea minimă Fmin=2 corespunde la x1=3 şi x2=9; aceasta a fost găsită după Nr=248 paşi de calcul.

13.3. Calculul zerourilor funcţiilor de o variabilă reală

Funcţia *fzero* se apelează cu una dintre sintaxele:

z=*fzero*('F',x0) z=*fzero*('F',x0,tol) z=*fzero*('F',x0,tol,trace)

Aceasta returnează zeroul funcţiei F(x) cel mai apropiat de estimarea iniţială x0, calculat cu eroarea relativă `tol`. Valoarea predefinită a lui `tol` este *eps*. 'F' este numele unui fişier M care conţine funcţia obiectiv al cărui zerou este căutat. Pentru a afişa rezultatul după fiecare iteraţie, argumentul opţional `trace` trebuie să fie diferit de zero.

Se reaminteşte că zerourile unui polinom pot fi calculate şi cu funcţia *roots*.

Exemplul 13.3.1. Să se determine zeroul funcţiei f(x)=sin(x), cel mai apropiat de x0=3.

Deoarece funcţia considerată este predefinită în MATLAB, rezultatul se obţine cu secvenţa:

```
x = fzero('sin', 3)
```

fiind:

x = 3.1416

Exemplul 13.3.2. Să se determine zeroul funcţiei: $f(x) = (x-3)^2 - 1$, cel mai apropiat de x0=1.5.

Se scrie fişierul-M:

```
function y = test(x)
y = (x-3).^2 - 1;
```

care se apelează apoi cu instrucţiunea:

```
z=fzero('test',1.5)
```

obţinându-se rezultatul:

z = 2

14. REZOLVAREA SISTEMELOR DE ECUAŢII LINIARE

14. 1. Descrierea matriceală a sistemelor de ecuaţii

Fie următorul sistem de trei ecuaţii cu trei necunoscute:

$$\begin{cases} 3x+2y-z=10 \\ -x+3y+2z=5 \\ x-y-z=-1 \end{cases}$$

Acesta poate fi scris matriceal sub una din formele :

AX=B sau YC=D

În prima formă, adică AX=B:

- A este matricea coeficienţilor necunoscutelor, cu dimensiunea 3x3. Coeficienţii aceleiaşi necunoscute sunt pe aceeaşi coloană;
- X este matricea necunoscutelor, cu dimensiunea 3x1;
- B este matricea termenilor liberi cu dimensiunea 3x1;

În cazul sistemului dat ca exemplu:

$$A=\begin{bmatrix} 3 & 2 & -1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad X=\begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad B=\begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

În a doua formă de scriere matriceală, adică YC=D:

- C este matricea coeficienţilor necunoscutelor, cu dimensiunea 3x3. Coeficienţii aceleiaşi necunoscute sunt pe aceeaşi linie; matricea C este transpusă matricei A;
- Y este matricea necunoscutelor, cu dimensinea 1x3; matricea Y este transpusa matricei X.
- D este matricea termenilor liberi, cu dimensiunea 1x3; matricea D este transpusa matricei B.

Pentru sistemul de mai sus, rezultă:

$$C=A^T=\begin{bmatrix} 3 & -1 & -1 \\ 2 & 3 & 1 \\ 1 & 2 & 1 \end{bmatrix}$$

$$Y=X^T=\begin{bmatrix} x & y & z \end{bmatrix} \qquad D=B^T=\begin{bmatrix} 10 & 5 & -1 \end{bmatrix}$$

14.2. Rezolvarea sistemelor prin împărţirea matricelor

Una dintre metodele de rezolvare a sistemelor de N ecuaţii cu N necunoscute constă în împărţirea matricelor. Astfel, rezolvarea ecuaţiei matriceale AX=B presupune împărţirea la stânga a matricelor:

AX=B ⇒ X=A\B

Pentru exemplul din subcapitolul 14.1 se utilizează secvenţa MATLAB:

```
A=[3 2 -1; -1 3 2; 1 -1 -1];
B=[10 5 -1]';
X=A\B
```

se obţine rezultatul:

X= [-2.000 5.000 6.000]

care este echivalent cu :

x=-2 y=-5 z=-6

Dacă sistemul de ecuaţii este descris cu ecuaţia matriceală YC=D, rezolvarea acestuia presupune împărţirea la dreapta:

YC=D ⇒ Y=D/C

Pentru exemplul dat, folosind secvenţa MATLAB:

```
C=[3 -1 1;2 3 -1;-1 2 -1];
D=[10 5 -1];
Y=D/C
```

se obţine rezultatul :

Y = [-2.0000 5.0000 -6.0000]

Împărţirea la stânga este mai rapidă; de exemplu pentru un sistem de 300 ecuaţii cu 300 necunoscute, ai cărui coeficienţi şi termeni liberi sunt numerele aleatoare, cu secvenţa MATLAB:

```
A=rand(300);b=rand(300,1);c=rand(1,300);
tic; X=A\b; ts=toc
tic; X=c/A; toc
```

s-au obţinut, pe un calculator PC486/66MHz, timpii : ts=6.81 sec. şi td=7.14 sec. în cele două variante de împărţire, la stânga şi la dreapta.

14.3. Rezolvarea sistemelor prin folosirea matricei inverse

Dacă sistemul a fost descris cu ecuaţia matriceală:

AX=B

prin multiplicare la stânga cu A^{-1} (matricea inversă a lui A), se obţine :

$$A^{-1}AX=A^{-1}B$$

deoarece, $A^{-1}A$-I, unde I este matricea unitate, rezultă :

$$X=A^{-1}B$$

În MATLAB, ultima relaţie se exprimă sub forma:

X=*inv*(A)*B

inv fiind funcţia MATLAB de calcul a matricei inverse. Pentru exemplul din subcapitolul 14.1, soluţia se obţine cu secvenţa:

```
A=[3 2 -1; -1 3 2; 1 -1 -1];
B=[10 5 -1]';
X=inv(A)*B
```

Dacă sistemul este descris cu ecuaţia matriceală:

YC=D

prin multiplicare la dreapta cu C^{-1} (matricea inversă a lui C), se obţine :

$$YCC^{-1}=DC^{-1}$$

rezultând:

$$Y = DC^{-1}$$

În MATLAB, ultima relaţie se exprimă sub forma :

Y=D*inv(C)

Cu secvenţa MATLAB:

```
C=[3 -1 1;2 3 -1;-1 2 -1];
D=[10 5 -1];
Y=D*inv(C)
```

se obţine rezultatul :

Y = [-2.0000 5.0000 -6.0000]

Vectorii X si Y au aceleaşi valori, cu precizarea însă că X este un vector coloană, iar Y este vector linie.

Nu există diferenţe semnificative între înmulţirea la stânga sau la dreapta cu matricea inversă; de exemplu cu secvenţa MATLAB:

```
A=rand(300);b=rand(300,1);c=rand(1,300);
tic; X=inv(A)*b; ts=toc
tic; X=c*inv(A); toc
```

pe un calculator PC 486/66 Mhz, s-au obţinut rezultatele : ts=20.70 sec. şi td = 20.82 sec. Se observă însă, că timpul de calcul este de circa trei ori mai mare prin metoda matricei inverse, faţă de metoda împărţirii matricelor.

15. CALCULE NUMERICE CU POLINOAME

În acest capitol se prezintă funcţiile MATLAB pentru evaluarea polinoamelor, calculul derivatei polinoamelor, generarea polinoamelor, când se cunosc rădăcinile, precum şi descompunerea acestora în fracţii simple. Funcţiile MATLAB pentru calcule cu polinoame sunt:

conv	Calculează produsul a două polinoame;
deconv	Calculează câtul şi restul împărţirii a două polinoame;
poly	Calculează coeficienţii unui polinom cu rădăcinile date;
polyval	Evaluează un polinom la valorile precizate ale variabilei;
polyder	Calculează derivata polinoamelor;
polyfit	Aproximează un set de date cu un polinom de grad n;
residue	Descompune în fracţii simple raportul a două polinoame;
roots	Calculează rădăcinile unui polinom.

Polinomul este o funcţie cu o singură variabilă, care poate fi exprimat sub următoarea formă generală :

$$f(x) = a_1x^N + a_2x^{N-1} + a_3x^{N-2} + ... + a_{N-1}x^2 + a_Nx + a_{N+1}$$

unde x este variabila, iar a_1, a_2, ..., a_{N+1} sunt coeficienţii polinomului. Gradul polinomului este egal cu cea mai mare valoare a exponentului.

În sintaxa MATLAB polinoamele sunt reprezentate cu un vector linie care conţine coeficienţii în ordinea descrescătoare a puterilor variabilei. Coeficienţii polinomului au indicii de la 1 la N+1. Spre exemplu, reprezentarea în MATLAB a polinomului:

$$h(x) = x^2 + 2x^2 + 0.5x - 6.5$$

este dată de vectorul:

h =[1 2 0.5 -6.5]

15.1. Evaluarea polinoamelor

În MATLAB există mai multe metode pentru evaluarea polinoamelor. Vor fi exemplificate, cu polinomul:

$$f(x) = 3x^4 - x^3 + x - 1$$

Cea mai simplă metodă este evaluarea cu scalari, adică pentru o singură valoare a variabilei. Se efectuează operaţiile cu scalari, ca în exemplul :

```
x=1;
f=3*x^4-x^3+x-1
```

obţinându-se rezultatul: f= 2

A doua metodă constă în evaluarea polinomului în mai multe puncte. În acest caz se efectuează operaţii cu tablouri. Programul se execută mai rapid decât ansamblul calculelor pentru fiecare valoare scalară în parte. Sintaxa evaluării presupune în acest caz plasarea unui punct înaintea operatorilor înmulţire, împărţire sau ridicare la putere. Dimensiunea matricei în care se returnează valorile polinomului este identică cu cea a matricei care conţine punctele în care se face evaluarea. Evaluarea polinomului f(x) în

punctele -2, 0 1, 2.5 3, se face cu secvenţa MATLAB:

```
x=[-2  0 1  2.5  3];
f=3*x.^4-x.^3+x-1 .
```

obţinându-se rezultatul:

f = [53.0000 -1.0000 2.0000 103.0625 218.0000]

A treia metodă de evaluare a polinoamelor constă în utilizarea funcţiei *polyval*, se apelează cu sintaxa:

f=*polyval*(p,s)

Funcţia *polyval* evaluează polinomul definit de vectorul p, al coeficienţilor polinomului, în punctul s. Dacă s este vector sau matrice, polinomul este evaluat pentru fiecare dintre elementele acestuia.

Pentru evaluarea polinomului luat ca exemplu, în punctele -2, 0 1, 2.5 3, secvenţa MATLAB este următoarea:

```
p=[3  -1  0  1  -1];
x=[-2  0 1  2.5  3];
f=polyval(p,x)
```

iar rezultatul obţinut este:

f = [53.0000 -1.0000 2.0000 103.0625 218.0000]

Pentru evaluarea unui polinom pe un interval, se generează în primul rând un vector al punctelor de evaluare, cu sau fără pas constant, şi apoi se procedează ca mai sus. Evaluarea polinomului f(x), definit anterior, în 300 puncte pe intervalul [0, 5], se poate face cu secvenţa MATLAB:

```
x=0:5/300:5;
p=[3  -1  0  1  -1];
f=polyval(p,x)
```

15.2. Adunarea şi scăderea

Fie două polinoame g(x) şi h(x) ai căror coeficienţi sunt înregistraţi în vectorii linie g şi h. Operaţiile aritmetice de adunare şi scădere a polinoamelor presupun adunarea şi scăderea coeficienţilor de acelaşi ordin. Deoarece în MATLAB aceste operaţii necesită vectori de aceeaşi dimensiune, lungimea vectorilor coeficienţilor este dată de polinomul cu puterea cea mai mare. Odată stabilită această dimensiune, spre exemplu M, vectorii coeficienţilor polinoamelor care au puterea maximă N mai mică decât M vor fi completaţi la stânga cu coeficienţi zero până la M. Spre exemplu, pentru:

$$g(x) = x^4 - 3x^2 - x + 2$$
$$h(x) = 2x^2 + 5x - 16$$
$$s(x) = g(x) + h(x)$$
$$u(x) = g(x) - h(x)$$

dimensiunea vectorului coeficienţilor va fi 5, iar polinoamele echivalente pentru care se scriu vectorii coeficienţilor sunt:

$$g(x) = x^4 + 0x^3 - 3x^2 - x + 2$$
$$h(x) = 0x^4 + 0x^3 + 2x^2 + 5x - 16$$

Secvenţa MATLAB care realizează operaţiile de mai sus este:

```
g=[1, 0, -3, -1,  2];
```



```
h=[0, 0,  2,  5, -16];
s=g+h
u=g-h
```

obţinându-se rezultatele:

s = [1 0 -1 4 -14]
u = [1 0 -5 -6 18]

respectiv polinoamele:

$$s(x) = x^4 - x^2 + 4x - 14$$
$$u(x) = x^4 - 5x^2 - 6x + 18$$

15.3. Înmulţirea şi împărţirea

Multiplicarea unui polinom cu un scalar constă în multiplicarea coeficienţilor acestuia cu acel scalar.

Înmulţirea a două polinoame este echivalentă unei operaţii de convoluţie; este realizată cu funcţia MATLAB *conv*, care se apelează cu sintaxa:

c = *conv*(a,b)

unde: a şi b sunt vectorii coeficienţilor polinoamelor care se înmulţesc;
c este vectorul coeficienţilor polinomului produs c(x)=a(x)b(x).

Împărţirea a două polinoame este echivalentă unei operaţii de deconvoluţie; este realizată cu funcţia MATLAB *deconv*, care se apelează cu sintaxa:

[d, r] = *deconv*(a,b)

unde: - a şi b sunt vectorii coeficienţilor polinoamelor deîmpărţit şi împărţitor;
- d este vectorul coeficienţilor polinomului cât;
- r este vectorul coeficienţilor polinomului rest, rezultat al operaţiei:

$$a(x) = d(x) \cdot b(x) + r(x)$$

Pentru ilustrarea modului de utilizare a funcţiilor *conv* şi *deconv* la înmulţirea şi împărţirea polinoamelor, fie calculele:

$$g(x) = (x^2 + x - 2)(x + 1)$$
$$h(x) = \frac{x^3 + 2x^2 - 2}{x^2 + 1}$$

Cu notaţiile:

$$g(x) = g1(x) \cdot g2(x)$$
$$h(x) = \frac{f(x)}{e(x)}$$

şi secvenţa MATLAB:

```
g1=[1  1  -2];  g2=[1  1];
f =[1  2  0  -2 ]; e =[1  0  1];
g= conv(g1,g2)
[h, r]=deconv(f,e)
```

se obţin rezultatele:

g = [1 2 -1 -2]
h = [1 2]
r = [0 0 -1 -4]

care reprezintă polinomul produs:

$$g(x)=x^3+2x^2-x-2$$

polinomul cât:

$$h(x)=x+2$$

şi restul:

$$r(x)=-x-4$$

15.4. Descompunerea în fracţii simple

Prin descompunere în fracţii simple se înţelege scrierea raportului a două polinoame ca sumă de fracţii cu polinoame de ordinul unu:

$$\frac{B(x)}{A(x)}=\frac{r(1)}{x-p(1)}+\frac{r(2)}{x-p(2)}+\ldots+\frac{r(n)}{x-p(n)}+k(x)$$

Dacă polul p(j) este de ordinul m, descompunerea conţine termeni de forma:

$$\frac{r(j)}{x-p(j)}+\frac{r(j+1)}{(x-p(j))^2}+\ldots+\frac{r(j+m-1)}{(x-p(j))^m}$$

Funcţia MATLAB *residue* se apelează cu sintaxa:

[r,p,k]=*residue*(B,A)

unde:

- A, B - vectorii linie ai coeficienţilor polinoamelor numitor şi numărător, în ordinea descrescătoare a puterilor variabilei;
- r - vectorul coloană al rezidurilor;
- p - vectorul coloană al polilor;
- k - vectorul linie al termenilor liberi.

Dacă funcţia *residue* se apelează cu sintaxa:

[B,A]=*residue*(r,p,k)

se returnează coeficienţii polinoamelor B şi A, numărător şi numitor, al căror raport are rezidurile r, polii p şi termenii liberi k.

Exemplul 15.4.1. Să se descompună în fracţii simple expresia:

$$\frac{B(x)}{A(x)}=\frac{x^3+2x^2-2}{x^2+1}$$

Cu secvenţa MATLAB:

```
B=[1  2  0  -2 ];
A=[1  0  1];
[r, p, k]=residue(B,A)
```

se obţin rezultatele:

r =	p =	k =
-0.5000 + 2.0000i	0 + 1.0000i	1 2
-0.5000 - 2.0000i	0 - 1.0000i	

care reprezintă următoarea descompunere în fracţii simple:

$$\frac{B(x)}{A(x)}=\frac{-0.5+2i}{x-i}+\frac{-0.5-2i}{x+i}+x+2$$

15.5. Calculul derivatei

Derivata unui polinom în MATLAB utilizează funcţia *polyder*, care se apelează cu sintaxa:

D=*polyder*(C)

unde C este vectorul linie al coeficienţilor polinomului, în ordinea descrescătoare a puterilor variabilei, iar D este vectorul linie al coeficienţilor polinomului derivat.

Apelată cu sintaxa:

D=*polyder*(A,B)

funcţia *polyder* returnează în vectorul D coeficienţii derivatei polinomului produs AB:

$$D(x)=\left(A(x)\cdot B(x)\right)'=A'(x)\cdot B(x)+A(x)\cdot B'(x)$$

Apelată cu sintaxa:

[M,N]=*polyder*(A,B)

funcţia *polyder* returnează în vectorii M şi N coeficienţii numărătorului şi numitorului derivatei raportului polinoamelor A/B:

$$\frac{M(x)}{N(x)}=\left(\frac{A(x)}{B(x)}\right)'=\frac{A'(x)\cdot B(x)-A(x)\cdot B'(x)}{(B(x))^2}$$

Exemplul 15.5.1. Fie polinoamele:

$$A(x)=x^2+2x-1$$

$$B(x)=x-1$$

Calculaţi derivatele polinoamelor:

$$Ad(x)=A'(x) \qquad Bd(x)=B'(x)$$

$$Cd(x)=\left(A(x)\cdot B(x)\right)' \qquad Dd(x)=\frac{M(x)}{N(x)}=\left(\frac{A(x)}{B(x)}\right)'$$

Cu secvenţa MATLAB:

```
A=[ 1   2   -1];   B=[ 1  -1];
Ad=polyder(A)
Bd=polyder(B)
Cd=polyder(A,B)
[M, N]=polyder(A,B)
```

se obţin rezultatele:

```
Ad = [  2   2 ]        Bd =   1
Cd = [  3   2  -3]
M  = [  1  -2  -1 ]
N  = [  1  -2   1 ]
```

care reprezintă polinoamele:

$$Ad(x)=2x+2 \qquad Bd(x)=1$$

$$Cd(x)=3x^2+2x+3 \qquad Dd(x)=\frac{M(x)}{N(x)}=\frac{x^2-2x-1}{x^2-2x+1}$$

15.6. Calculul rădăcinilor

În acest subcapitol sunt prezentate funcţiile MATLAB pentru calculul rădăcinilor unui polinom sau pentru calculul coeficienţilor unui polinom atunci când se cunosc rădăcinile acestuia.

Dacă funcţia f(x) este un polinom de gradul N, atunci f(x)=0 are N rădăcini, care pot fi reale sau complexe. În cazul în care coeficienţii polinomului sunt numere reale, rădăcinile complexe sunt în perechi complex conjugate.

Funcţia MATLAB *roots* determină rădăcinile polinoamelor; se apelează cu sintaxa:

r=*roots*(c)

Funcţia MATLAB *poly* determină coeficienţii unui polinom ale cărui rădăcini sunt cunoscute; se apelează cu sintaxa:

c=*poly*(r)

În ambele cazuri:

c - este un vector linie, care conţine coeficienţii polinomului, în ordine descrescătoare a puterilor variabilei;

r - este un vector coloană care conţine rădăcinile polinomului.

Funcţiile *roots* şi *poly* sunt funcţii reciproce, realizând trecerea de la coeficienţi la rădăcini sau de la rădăcini la coeficienţi.

Exemplul 15.6.1. Determinaţi rădăcinile polinomului:

$$f(x) = x^3 - 2x^2 - 3x + 10$$

şi efectuaţi verificarea rezultatului.

Cu secvenţa MATLAB:

```
p=[1, -2, -3, 10]; r=roots(p)
```

se obţine rezultatul:

```
r =
  2.0000 + 1.0000i
  2.0000 - 1.0000i
 -2.0000
```

Cu secvenţa MATLAB:

```
c=[1, -2, -3, 10]; %coeficientii
r=[ 2+i  2-i  -2]; %radacinile
v=polyval(c,r)
```

se obţine rezultatul v=[0 0].

Exemplul 15.6.2. Să se determine coeficienţii polinomului care are rădăcinile r=[-1, 1, 3].

Cu secvenţa MATLAB:

```
c=poly([-1, 1, 3]')
```

se returnează vectorul:

c= [1 -3 -1 3]

care reprezintă polinomul:

$$f(x) = x^3 - 3x^2 - x + 3$$

16. INTERPOLAREA ŞI APROXIMAREA DATELOR

Funcţiile MATLAB pentru interpolarea şi aproximarea datelor sunt:

table1	Interpolează liniar şi citeşte date din tabele unidimensionale;
spline	Interpolează prin metoda spline date din tabele unidimensionale;
interpft	Interpolează prin metoda transformatei Fourier datele din tabele unidimensionale;
polyfit	Aproximează un set de date cu un polinom de gradul n;
interp1	Interpolează liniar, spline sau cubic date din tabele unidimensionale;
table2	Interpolează liniar şi citeşte date din tabele bidimensionale;
interp2	Interpolează biliniar sau bicubic date din tabele bidimensionale;
interp3	Interpolează biarmonic valorile unei funcţii de două variabile;
interp4	Interpolează biliniar valorile unei funcţii de două variabile;
interp5	Interpolează bicubic valorile unei funcţii de două variabile;
griddata	Interpolează prin metoda distanţei inverse valorile unei funcţii de două variabile.

Interpolarea unui set discret de date $[x_i, y_i]$ presupune determinarea unei funcţii f(x) astfel ca $f(x_i)=y_i$, în vederea completării setului de date în orice alt punct $x_0 \neq x_i$. Spre exemplu, fie date punctele de coordonate (x_1,y_1) şi (x_2,y_2). Se cere estimarea valorii y(x), unde $x_1<x<x_2$. Dacă punctele sunt unite printr-o dreaptă, interpolarea se numeşte liniară, iar dacă sunt unite printr-un polinom de gradul trei, interpolarea este spline cubică.

O altă problemă constă în aproximarea unui set de date cu o funcţie care constituie „cea mai bună aproximare". În acest caz nu este neapărat necesar ca funcţia determinată să treacă prin toate punctele date, însă trebuie să fie „cea mai bună aproximare" într-un anumit sens (după un anumit criteriu de eroare impus). Metoda celor mai mici pătrate furnizează de exemplu cea mai bună aproximare în sensul minimizării pătratului distanţelor dintre punctele date şi funcţia de aproximare.

16.1. Căutarea datelor în tabele

Prin căutarea datelor în tabele se înţelege operaţia prin care o anumită valoare este citită dintr-un tabel predefinit. Aceste operaţii sunt utile în proiectare, când trebuie căutate elemente normalizate de dimensiuni sau parametri ai unor dispozitive, de exemplu alegerea unei secţiuni de conductor, alegerea unui dispozitiv electronic cu anumiţi parametri, dimensiuni mecanice ale unui profil etc.

16.1.1. Căutarea datelor în tabele unidimensionale

Căutarea datelor în tabele unidimensionale se face cu funcţia *table1*; se apelează cu sintaxa:

z=*table1*(tab,x)

Tabelul „tab" din care se citesc datele trebuie să fie organizat ca o matrice care are în prima

coloană valorile lui x (abscisa ordonată crescător), iar în coloanele 2:n+1 cele n variabile căutate. Dacă valoarea x se găseşte între două valori din prima coloană a tabelului „tab", funcţia returnează o valoare interpolată liniar.

Exemplul 16.1.1. Să se citească valorile y care corespund lui x1=-1 şi x2=3, din tabelul unidimensional de mai jos:

x	-2	-1	0	2	4
y	6	2	3	1	7

Se creează matricea coloană M, care are ca primă coloană elementele lui x ordonate crescător, iar a doua coloană elementele lui y în ordinea de asociere la x. Citirea din acest tabel a valorilor y1 şi y2, ce corespund lui x1=-1 şi x2=3, se face cu instrucţiunile:

```
M = [-2 -1  0  2  4; 6  2  3 -1  7 ];
M=M';
y1=table1(M,-1)
y2=table1(M,3)
```

obţinându-se rezultatele:

y1 = 2 y2 = 3

Observaţie: Pentru a utiliza funcţia *table1*, prima coloană a tabelului unidimensional din care se citesc datele trebuie să fie monotonă, iar valoarea lui x trebuie să se afle între prima şi ultima valoare din prima coloana a acestuia (între x_{min} şi x_{max}).

16.1.2. Căutarea datelor în tabele bidimensionale

Căutarea datelor în tabele bidimensionale se face cu funcţia *table2*; se apelează cu sintaxa:

z=*table2*(tab,x,y)

Tabelul bidimensional „tab" din care se citesc datele trebuie să fie organizat ca o matrice care are în prima coloană valorile x (ordonate crescător), iar în prima linie valorile y (ordonate crescător). Celelalte elemente din tabel sunt valorile z asociate perechilor (x,y). Elementul z(1,1) = 0.

Dacă valoarea x sau y se găseşte între două valori din prima coloană sau linie a tabelului „tab", funcţia returnează o valoare interpolată liniar.

Exemplul 16.1.2. Să se citească valorile care corespund perechilor (x,y); (2, 3), (2.5, 3) şi (2.5, 3.5), din următorul tabel bidimensional:

x\y	1	2	3	4
1	2	3	4	5
2	3	4	5	6
3	4	5	6	7

Se creează mai întâi matricea N care constituie tabelul din care se citeşte:

- elementul primei linii din prima coloană:
  N(1,1) = 0
- prima linie, mai puţin primul element, conţine elementele lui y ordonate crescător:

  N(1,2:n+1) = y(1:n).
- prima coloană, mai puţin primul element, vor fi elementele lui x ordonate crescător:
  N(2:n+1,1) = x(1:n).
- celelalte elemente ale matricei, sunt elementele asociate perechilor (x,y).

Prin urmare matricea N este:

$$N = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

Cu următoarea secvenţă MATLAB:

```
N=[0,1,2,3,4;1,2,3,4,5;2,3,4,5,6;3,4,5,6,7];
c1=table2(N, 2, 3)
c2=table2(N, 2.5, 3)
c3=table2(N, 2.5, 3.5)
```

se obţin rezultatele:

c1 = 5 c2 = 5.5000 c3 = 6

Observaţie: Pentru a utiliza funcţia *table2*, prima linie şi prima coloană a tabelului bidimensional din care se citesc datele trebuie să fie monotone. Valorile utilizate pentru x trebuie să se afle între prima şi ultima valoare din prima coloană (între x_{min} şi x_{max}) iar cele pentru y trebuie să se afle între prima şi ultima valoare din prima linie (între y_{min} şi y_{max}).

16.2. Interpolarea funcţiilor de o singură variabilă

În acest subcapitol se prezintă trei tipuri de interpolări: liniară, spline cubică şi polinomială. În toate cazurile se presupune că există un set de date care reprezintă coordonatele (x_i,y_i). Obiectivul îl constituie estimarea valorilor funcţiei f(x), pentru orice punct $x \in [x_1, x_2]$. Curba de interpolare trece prin toate punctele care o definesc, legea de interpolare între puncte putând fi liniară, cubică sau polinomială. În figura 16.1 este dat un exemplu grafic de interpolare liniară şi spline cubică a datelor. Valorile estimate prin interpolare între punctele date depind de tipul de interpolare ales.

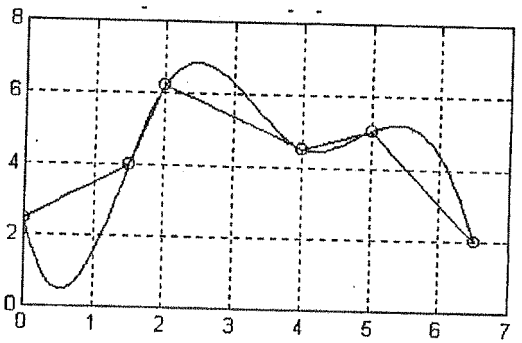


Fig. 16.1. Interpolare liniară şi spline cubică.

16.2.1. Interpolarea liniară

Dacă se presupune că funcţia dintre cele două puncte din figura 16.2 poate fi estimată printr-o linie dreaptă, atunci valoarea funcţiei în orice punct x dintre cele două valori se deduce cu expresia:

$$f(x) = y_1 + \frac{x - x_1}{x_2 - x_1}(y_2 - y_1)$$

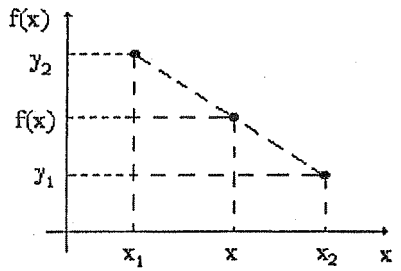


Fig. 16.2. Interpolarea liniară

Interpolarea liniară a funcţiilor de o singură variabilă se face cu funcţia *table1* care se apelează cu sintaxa:

y=*table1*(nume_tabel,x)

Primul argument al funcţiei este numele tabelului care conţine datele ce se referă la coordonatele (x_i,y_i). Dacă acesta este un fişier pe disc, atunci nume_fişier trebuie să fie mai întâi încărcat cu funcţia *load*.

Al doilea argument se referă la valorile lui x pentru care se doreşte determinarea valorilor interpolate y. Datele din prima coloană a tabelului (valorile lui x) trebuie să fie în ordine crescătoare, iar valorile x trebuie să se găsească între prima şi ultima valoare a primei coloane; în caz contrar se afişează un mesaj de eroare.

Dacă tabelul din care se citesc datele conţine mai mult de două coloane, funcţia *table1* returnează un vector linie cu N-1 elemente unde N este numărul de coloane ale tabelului. Fiecare valoare returnată este interpolată din coloana corespunzătoare a datelor.

Exemplul 16.2.1. Estimaţi valorile temperaturii la momentele de timp 2.5 sec şi 4.9 sec, cu datele din tabelul de mai jos.

Timp [s]	Temperatură [°C]
0. 0	0. 0
1. 0	20. 0
2. 0	60. 0
3. 0	68. 0
4. 0	77. 0
5. 0	110. 0

Se introduc datele din tabel într-o matrice care are în prima coloană valorile timpului şi în a doua coloană temperaturile corespunzătoare:

```
templ(:,1)=[0.0,1.0,2.0,3.0,4.0,5.0]' ;
templ(:,2)=[0.0,20.0,60.0,68.0,80.0,110.0]';
```

Cu instrucţiunea MATLAB:

```
y=table1(templ,[2.5  4.9])
```

se obţin rezultatele:

y= [64.0 107.0] °C

16.2.2. Interpolarea spline cubică

Curba spline cubică este o curbă netedă, definită de un set de polinoame de gradul trei. Curba dintre fiecare pereche de puncte este un polinom de gradul trei, calculat astfel încât să conducă la tranziţii netede de la un polinom de gradul trei la altul. Spre exemplu, şase puncte sunt conectate cu cinci curbe diferite de gradul trei, ce constituie o funcţie netedă între toate cele şase puncte.

Curba de interpolare spline cubică este calculată cu funcţia *spline*; se apelează cu sintaxa:

yi = *spline*(x,y,xi)

unde:

x şi y sunt vectorii care conţin abscisele şi ordonatele datelor (cu pas „mare");
xi este un vector care conţine noile abscise, de regulă cu pas mai „fin";
yi este vectorul returnat, asociat lui xi.

Valorile x trebuie să fie ordonate crescător.

Exemplul 16.2.2. Să se determine valoarea y de interpolare spline cubică pentru x = 2.6.

Cu secvenţa MATLAB:

```
x=[0, 1, 2, 3, 4, 5];
y= [0.0, 20.0, 60.0, 68.0, 77.0, 110.0]
tempa=spline(x,y, 2.6)
```

se obţine rezultatul:

tempa= 67.3013 °C

Pentru calculul a două valori interpolate, se poate utiliza secvenţa MATLAB:

```
tempb=spline(x,y,[2.6, 4.9])
```

obţinându-se rezultatul:

tempb= [67.3013 105.2020] °C

Exemplul 16.2.3. Să se reprezinte grafic o interpolare liniară şi spline cubică prin şase puncte. Cu secvenţa MATLAB:

```
x=[0, 1, 2, 3, 4, 5];
y= [0.0,20.0,60.0,68.0,77.0,110.0];
xi=0:0.1:5;
yi=spline(x,y,xi);
axis([-1, 6, -20, 120])
plot(x,y,xi,yi,x,y,'o')
title('Interpolarea liniara si spline')
```

```
xlabel('Timp [s]');
ylabel ('Temperatura [grade C]'); grid
```

se obţine graficul din figura 16.3.

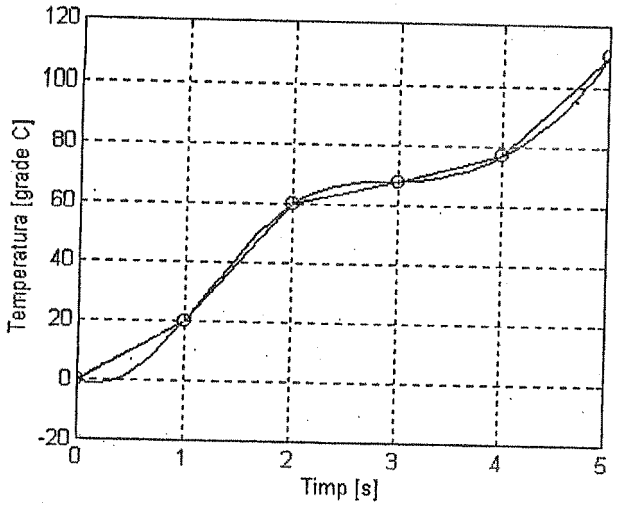


Figura 16.3. Interpolarea liniară şi spline a datelor

Exemplul 16.2.4. Fiind date 10 perechi de puncte în planul xOy, să se traseze graficul rezultat prin interpolarea *spline* pe o reţea de 4 ori mai fină.
Cu secvenţa de program:

```
x = 0:10; y = sin(x);
xi = 0:.25:10;
yi = spline(x,y,xi);
plot(x,y,'o',xi,yi)
```

se determină vectorul yi al celor 40 de puncte corespondente valorilor xi.

16.2.3. Interpolarea prin metoda transformatei Fourier

Funcţia *interpft* interpolează datele cu o singură variabilă utilizând metoda FFT (Fast Fourier Transform); se apelează cu sintaxa:

y=*interpft*(x,n)

care returnează un vector y de lungime n obţinut din vectorul x. Numărul n trebuie să fie mai mare decât numărul de elemente al vectorului x, iar rezultatul are periodicitate circulară dată de utilizarea transformatei Fourier.

Exemplul 16.2.5. Fie datele obţinute dintr-un sinus eşantionat cu 8 paşi pe perioadă:

$$y = \sin\left(\frac{2\pi k}{8}\right)$$

Interpolaţi datele cu un pas dublu şi verificaţi valorile obţinute.

Cu secvenţa MATLAB:

```
k=0:7;
x=sin(2*pi*k/8);
yi=interpft(x,16);
k1=0:15;
yr= sin(2*pi*k1/16);
d=max(yi-yr)
```

se obţine rezultatul: d = 5.5511e-016
Diferenţa maximă între valorile interpolate şi cele reale este egală cu ordinul de mărime al celui mai mic număr reprezentabil în calculator (eroarea de trunchiere).

16.2.4. Interpolarea multiplă

Funcţia *interp1* poate interpola datele după o lege liniară, cubică sau spline cubică; se apelează cu sintaxa:

yi=*interp1*(x,y,xi, 'regula')

unde „regula" este unul din cuvintele cheie:

linear - pentru interpolare liniară;
spline - pentru interpolare spline cubică;
cubic - pentru interpolare cubică.

Toate legile de interpolare necesită ca x să fie ordonat monoton. În plus, metoda 'cubic' cere ca punctele pe axa x să fie egal distanţate.

Dacă y este o matrice cu numărul de linii egal cu cel al vectorului x şi cu N coloane, *interp1* returnează o matrice cu un număr de linii egal cu cel al vectorului xi şi cu N coloane.

Exemplul 16.2.6. . Fie următoarele date ale unei funcţii de o singură variabilă:

x	-2	-1	0	1	2
y	6	2	3	1	7

Să se estimeze cu funcţia *interp1* valoarea y care corespunde lui x1=-1 şi x2=1.5, folosind interpolarea liniară, cubică şi spline cubică.
Cu secvenţa MATLAB:

```
x=[-2 -1 0 1 2];
y=[ 6  2 3 1 7];
yil=interp1(x,y,[-1,1.5],'linear')
yis=interp1(x,y,[-1,1.5],'spline')
yic=interp1(x,y,[-1,1.5],'cubic')
```

se obţin rezultatele:

yil = [2 4] - interpolare liniară
yis = [2.0000 2.0156] - interpolare spline
yic = [2 3] - interpolare cubică

16.3. Interpolarea funcţiilor de două variabile

Funcţiile MATLAB pentru interpolarea datelor care reprezintă funcţii de două variabile (suprafeţe) sunt *interp3*, *interp4*, *interp5* şi *griddata*.

16.3.1. Interpolarea după o lege biarmonică

Funcţia *interp3* interpolează datele funcţie de două variabile după o lege biarmonică; se apelează cu sintaxa:

zi=*interp3*(x,y,z,xi,yi)

care returnează în matricea zi valorile interpolate corespunzătoare lui xi şi yi. Matricele x şi y specifică punctele în care sunt date valorile lui z.

Dacă x şi y sunt vectori, punctele trebuie să fie egal distanţate şi monotone.

Apelată cu sintaxa:

zi=*interp3*(z,m,n)

funcţia returnează o matrice zi cu dimensiunea m x n, care reprezintă interpolarea datelor din matricea z.

16.3.2. Interpolarea după o lege biliniară

Funcţia *interp4* interpolează datele funcţie de două variabile după o lege biliniară; se apelează cu sintaxa:

zi=*interp4*(x,y,z,xi,yi)

care returnează în matricea zi valorile corespunzătoare lui xi şi yi. Matricele x şi y specifică punctele în care sunt date valorile lui z.

Dacă x şi y sunt vectori, punctele trebuie să fie egal distanţate şi monotone. Valori în afara intervalelor de definiţie pentru x şi y, returnează NaN pentru zi.

Apelată cu sintaxa:

zi=*interp4*(z,xi,yi)

funcţia presupune x=1:n şi y=1:m, unde m x n este dimensiunea matricei z.

Dacă se foloseşte sintaxa:

zi=*interp4*(z,k)

funcţia returnează o matrice zi care expandează matricea z prin întreţeserea unei coloane şi a unei linii interpolate bilinear între cele existente, operând recursiv de k ori. Prin urmare, pentru o matrice z cu dimensiunea 3x4 şi pentru k=2, rezultă la prima interpolare matricea zi cu dimensiunea 5x7, iar după a doua interpolare o matrice cu dimensiunea 9x13. Prin acest procedeu pasul devine de $p=2^k$ ori mai fin.

Spre deosebire de funcţiile care citesc din tabele prin interpolare valorile precizate, această funcţie returnează o matrice a tuturor valorilor interpolate.

Exemplul 16.3.1. Fie matricea: $A=\begin{bmatrix} 2 & 6 & 6 & 5 \\ 3 & 4 & 8 & 6 \\ 4 & 6 & 9 & 5 \end{bmatrix}$.

Interpolaţi biliniar matricea A folosind întreţeserea cu k=2 nivele de date interpolate.

Verificaţi dacă valoarea interpolată care corespunde la jumătatea intervalului dintre ultimele două coloane şi ultimele două linii după prima interpolare este media aritmetică a numerelor care definesc dreptunghiul de interpolare, adică (8+6+9+5)/4=7.

Cu secvenţa MATLAB:

```
A=[ 2,6,6,5;3,4,8,6;4,6,9,5]
Ai=interp4(A,2)
A1= interp4(A,1);
V=A1(4, 6)
```

se obţine matricea Ai care interpolează de două ori matricea A, A1 care interpolează o singură dată elementele matricei A şi valoarea cerută prin enunţ V=A1(4,6)=7.

16.3.3. Interpolarea după o lege bicubică

Funcţia *interp5* interpolează datele funcţie de două variabile după o lege bicubică; se apelează cu sintaxa:

zi=*interp5*(x,y,z,xi,yi)

care returnează în matricea zi valorile corespunzătoare lui xi şi yi. Matricele x şi y specifică punctele în care sunt date valorile lui z.

Dacă x şi y sunt vectori, punctele trebuie să fie egal distanţate şi monotone. Valori în afara intervalelor de definiţie pentru x şi y returnează NaN pentru zi.

Apelată cu sintaxa:

zi=*interp5*(z,xi,yi)

funcţia presupune x=1:n şi y=1:m, unde m x n este dimensiunea matricei z.

Dacă se foloseşte sintaxa:

zi=*interp5*(z,k)

funcţia returnează o matrice zi care expandează matricea z prin întreţeserea unei coloane şi a unei linii interpolate bicubic între cele existente, operând recursiv de k ori. Prin urmare, pentru o matrice z cu dimensiunea 3x4 şi pentru k=2, rezultă la prima interpolare matricea zi cu dimensiunea 5x7, iar după a doua interpolare o matrice cu dimensiunea 9x13. Prin acest procedeu pasul devine de $p=2^k$ ori mai fin.

Exemplul 16.3.2. Fie matricea: $A=\begin{bmatrix} 2 & 6 & 6 & 5 \\ 3 & 4 & 8 & 6 \\ 4 & 6 & 9 & 5 \end{bmatrix}$

Interpolaţi bicubic matricea A folosind întreţeserea cu k=2 nivele de date interpolate. Calculaţi valoarea interpolată care corespunde la jumătatea intervalului dintre ultimele două coloane şi ultimele două linii.

Cu secvenţa MATLAB:

```
A=[ 2,6,6,5;3,4,8,6;4,6,9,5]
Ai=interp5(A,2)
A1= interp5(A,1);
V=A1(4,6)
```

se obţine matricea Ai care interpolează de două ori matricea A, matricea A1 care interpolează o singură dată matricea A şi valoarea cerută prin enunţ V=A1(4,6)=8.06257. Rezultatul obţinut cu funcţia *interp5* este diferit de acela obţinut prin interpolarea biliniară.

16.3.4. Interpolarea prin metoda distanţei inverse

Interpolarea datelor funcţie de două variabile prin metoda distanţei inverse se realizează cu funcţia *griddata*; se apelează cu sintaxa:

zi=*griddata*(x,y,z,xi,yi)

Valorile matricei zi corespund elementelor matricelor xi şi yi. Matricea z asociază fiecărei perechi (x_i,y_j) câte o valoare $z_{i,j}$.

Dacă x este un vector linie, acesta se aplică fiecărei coloane a matricei z (este o matrice cu coloane constante). Similar, dacă y este un vector coloană, acesta se aplică fiecărei linii a matricei z (este o matrice cu linii constante).

Apelată cu sintaxa:

[xi, yi, zi]=*griddata*(x, y, z, xi, yi)

funcţia *griddata* returnează şi matricele xi şi yi.

Exemplul 16.3.3. Să se reprezinte grafic funcţia de două variabile $z=x(x^{-2}+y^{-2})$ pe intervalul $x\in[-1, 1]$, $y\in[-1, 1]$.

Cu secvenţa MATLAB:

```
x=-1:.5:1;  y=-1:.5:1;y=y';
[x,y]=meshgrid(x,y)
z=x.*(x.^2+y.^2);
xi=-1:.1:1; yi=-1:.1:1;
[xi,yi]=meshgrid(xi,yi)
zi=griddata(x,y,z,xi,yi)
mesh(x,y,z); hold on; mesh(xi,yi,zi-5)
```

se obţine reprezentarea din figura 16.4. (Pentru comparaţie rezultatul interpolării a fost deplasat pe axa z cu d=-5 faţă de reprezentarea neinterpolată.)

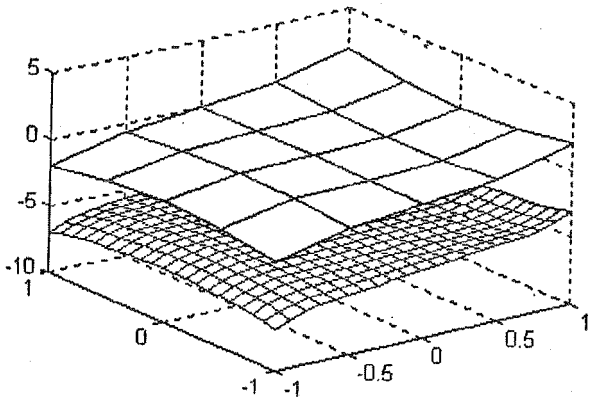


Fig. 16.4. Interpolarea suprafeţelor cu funcţia *griddata*.

16.3.5. Interpolarea multiplă

Funcţia *interp2* interpolează datele funcţiilor de două variabile după lege biliniară sau bicubică; se apelează cu sintaxa:

zi=*interp2*(x,y,z,xi,yi, 'regula')

unde „regula" este unul dintre cuvintele cheie:

linear - pentru interpolare biliniară;

cubic - pentru interpolare bicubică.

Matricea zi conţine valorile interpolate corespunzătoare lui xi şi yi.

Ambele legi de interpolare necesită ca x şi y să fie ordonate monoton. În plus, metoda 'cubic' cere ca punctele pe axele x şi y să fie egal distanţate.

Dacă x este un vector linie, acesta se aplică fiecărei coloane a matricei z. Similar, dacă y este un vector coloană, acesta se aplică fiecărei linii a matricei z.

Exemplul 16.3.4. Fie următorul tabel bidimensional:

y\x	1	2	3	4
1	2	6	6	5
2	3	4	8	6
3	4	6	9	5

Să se estimeze valorile zi care corespund perechii (x,y)= (3.5, 2.5), cu funcţia *interp2*, folosind interpolarea biliniară şi bicubică.

Cu secvenţa MATLAB:

```
x=[1 2 3 4];%vectorul liniilor constante
y=[1 2 3]'; %vectorul coloanelor const.
z=[2,6,6,5;3,4,8,6;4,6,9,5];
zil=interp2(x,y,z,3.5,2.5,'linear')
zic=interp2(x,y,z,3.5,2.5,'cubic')
```

se obţin rezultatele:

zil= 7 zic= 8.0625

16.4. Aproximarea prin metoda celor mai mici pătrate

În acestă secţiune se prezintă mai întâi aproximarea unui set de date printr-o linie dreaptă şi apoi aproximarea printr-un polinom. Pentru ca aproximarea să fie considerată „cea mai bună" suma pătratelor distanţelor de la fiecare punct la curba aproximantă (linie sau polinom) trebuie să fie minimă. Cu această condiţie este posibil ca nici un punct al setului de date să nu se găsească pe curba aproximantă. Acest lucru separă net aproximarea de interpolare, la care toate punctele sunt situate pe curbă.

16.4.1. Regresia liniară

Regresia liniară este aproximarea unui set de date printr-o dependenţă liniară care minimizează suma pătratelor dintre dreapta de aproximare şi punctele date.

Măsura calităţii unei aproximări liniare este dată de suma pătratelor distanţelor de la fiecare punct la estimaţia liniară, distanţe prezentate în figura 16.6, mărime dată de expresia:

```
sum_p=sum((y-y1).^2)
```

Determinarea parametrilor m şi n ai dreptei de aproximare y=mx+n se face utilizând funcţia *polyfit*.

Exemplul 16.4.1. Să se aproximeze în sensul celor mai mici pătrate cu o regresie liniară setul de date:

x=[0, 1, 2, 3, 4, 5];
y=[0, 20, 60, 68, 77, 110];

Cu următoarea secvenţă MATLAB:

```
x=[0, 1, 2, 3, 4, 5];
y=[0, 20, 60, 68, 77, 110];
coef=polyfit (x, y, 1);
m = coef(1);
n=coef (2);
y1=m*x+n;
sum_p=sum((y-y1).^2)
axis([-1, 6, -20, 120])
plot(x, y1, x, y, 'o')
grid
```

se obţine graficul din figura 16.5.

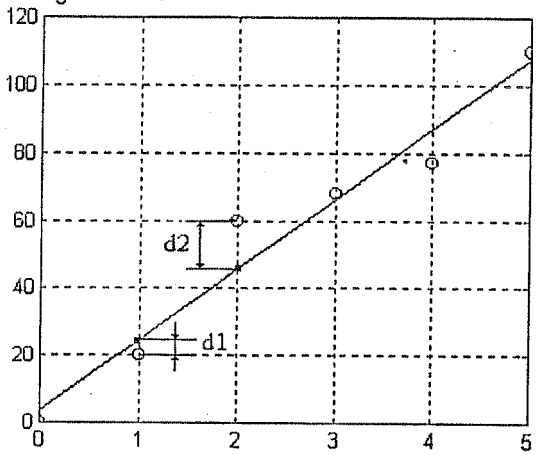


Fig.16.5. Aproximarea prin regresie liniară

16.4.2. Regresia polinomială

Regresia polinomială este o aproximare a unui set de date printr-un polinom de forma:

$$p(x) = \sum_{i=0}^{N} a_i x^{N-i} = a_0 x^N + a_1 x^{N-1} + \ldots + a_{N-1} x + a_N$$

Dacă setul de date are N elemente, toate datele se află pe curba de „aproximare". Pentru un grad al polinomului mai mic decât numărul de date, aproximarea este cu atât mai bună cu cât gradul polinomului este mai apropiat de numărul de date. Utilizarea unui polinom de aproximare cu grad mai mare decât setul de date poate conduce la erori de aproximare considerabile.

Determinarea celei mai bune aproximări a unui set de date (x,y) cu un polinom de ordin n foloseşte funcţia *polyfit*, se apelează cu sintaxa:

p=*polyfit*(x,y,n)

Funcţia *polyfit* returnează coeficienţii a_i ai polinomului p(x), care în punctele precizate de vectorul x are, în sensul celor mai mici pătrate, valorile date de vectorul y.

Exemplul 16.4.2. Fie polinomul $p(x) = x^3-6x^2+11x-6$, peste care este suprapus un zgomot cu distribuţie normală. Aproximaţi în sensul celor mai mici pătrate datele rezultate cu un polinom de gradul 3. Reprezentaţi grafic datele cu zgomot şi polinomul aproximant.

Cu secvenţa MATLAB:

```
p=[1, -6, 11, -6];
x=0:.25:4;
y=polyval(p,x)+randn(size(x));
c=polyfit(x,y,3);
poli3=polyval(c,x);
plot(x,poli3,x,y,'o');grid
```

se obţine graficul din figura 16.6.

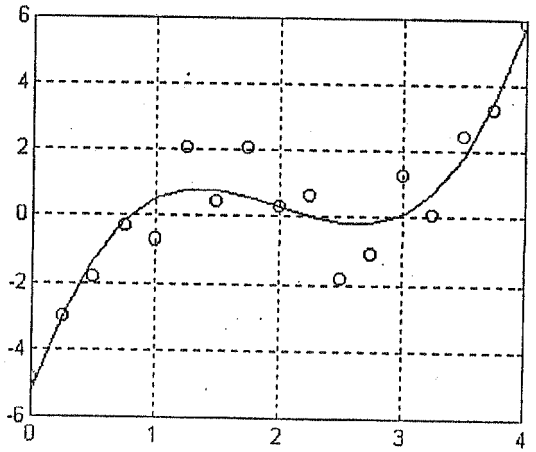


Fig. 16.6. Aproximarea datelor cu zgomot cu un polinom de gradul 3.

Exemplul 16.4.3. Să se determine coeficienţii polinoamelor de grad 3 şi 5 care aproximează punctele (x,y) precizate în tabelul de mai jos:

x	-2	-1	0	2	4
y	-15	-3	2	-3	10

Cu următoarea secvenţă de program:

```
x = [ -2 -1  0  2   4];
y = [-15 -3  2 -3  10];
xn = -2:.1:4;
c3 = polyfit(x,y,3)
y3 = polyval(c3,xn);
c5 = polyfit(x,y,5)
y5 = polyval(c5,xn);
```

se obţin coeficienţii:

c3 = [0.7719 -2.5526 0.0439 1.2105]

c5 = [-0.0958 0.4042 1.1333 -4.3667 0 2.0000]

care reprezintă polinoamele:

$$y3(x) = 0.7719x^3 - 2.5526x^2 + 0.0439x + 1.2105$$

$$y5(x) = -0.0958x^5 + 0.4042x^4 + 1.1333x^3 - 4.3667x^2 + 2$$

Exemplul 16.4.4. Să se scrie o secvenţă de program care să aproximeze prin polinoame de gradul 2, 5 şi 7 şi prin funcţii spline 5 perechi de puncte de coordonate cunoscute. Să se reprezinte grafic aproximările determinate.

Cu secvenţa MATLAB:

```
x = [ -2  -1  0 2 4 ];
y = [-19  -3  1  -3  17 ];
c2 = polyfit(x,y,2); % coef. polin.de grad 2
c5 = polyfit(x,y,5); % coef. polin.de grad 5
c7 = polyfit(x,y,7); % coef. polin.de grad 7
x1 = -2:.1:4;  % noul pas de interpolare
y2 = polyval(c2,x1); % eval. polinom de grad 2
y5 = polyval(c5,x1); % eval. polinom de grad 5
y7 = polyval(c7,x1); % eval. polinom de grad 7
yi = spline(x,y,x1);  % interpoleaza spline
plot(x1,yi,x1,y2,x1,y5,x1,y7,x,y,'o')
grid
```

Se obţine graficul din figura 16.7.

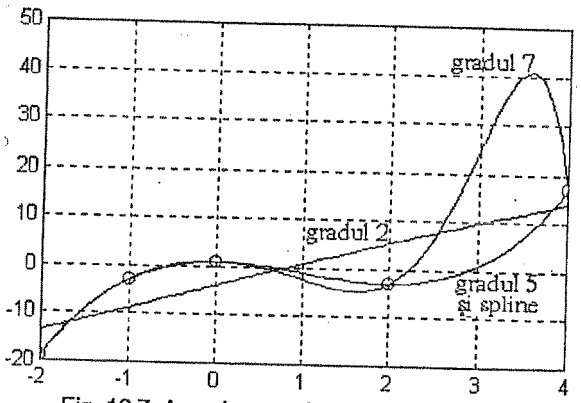


Fig. 16.7. Aproximarea datelor cu polinoame.

17. INTEGRAREA ŞI DERIVAREA NUMERICĂ A FUNCŢIILOR

Integrarea şi derivarea sunt concepte fundamentale în rezolvarea unui mare număr de probleme în inginerie şi în ştiinţă. În multe situaţii nu pot fi obţinute soluţii analitice, fiind necesară aplicarea metodelor de integrare şi derivare numerică. În acest capitol se prezintă funcţiile MATLAB pentru integrarea şi derivarea numerică.

17.1. Integrarea numerică

Integrala funcţiei f(x) pe intervalul [a,b]:

$$S=\int_a^b f(x)dx$$

are semnificaţia ariei delimitată de axa Ox, curba f(x) şi dreptele x=a şi x=b, figura 17.1

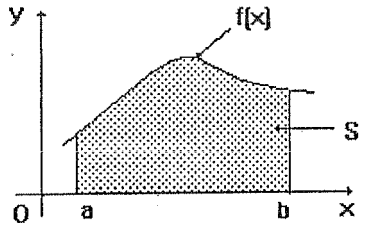


Fig 17.1. Integrala funcţiei f(x), de la a la b.

Calculul numeric al unei integrale denumit şi cuadratură se poate face prin:

- aproximarea funcţiei de integrat f(x) printr-o altă funcţie g(x), în care g(x) este astfel încât integrala să poată fi uşor calculată. O bună aproximare a funcţiei f(x) conduce la o bună estimare a integralei;
- aproximarea funcţiei f(x) cu un set de funcţii liniare sau parabolice, pe porţiuni. Dacă aproximarea se face cu funcţii liniare pe porţiuni, aria se poate calcula ca sumă a trapezelor care o compun, această metodă fiind cunoscută ca metoda trapezelor. Dacă aproximarea se face cu funcţii pătratice pe porţiuni, metoda este cunoscută ca metoda lui Simpson.

Integrarea numerică atenuează erorile de aproximare a funcţiei.

17.1.1. Metoda trapezelor şi metoda lui Simpson

Dacă integrala se aproximează prin trapeze, iar intervalul [a,b] este împărţit în n secţiuni egale, adică pasul şi abscisa curentă sunt:

$$h=\frac{b-a}{n}>0 \qquad x_j=a+jh \qquad 1\le j\le n$$

integrala poate fi estimată prin expresia:

$$S_T=\frac{h}{2}\left[y_a+y_b+2\sum_{j=1}^{n-1}y_j\right]$$

unde $y_j=f(x_j)$.

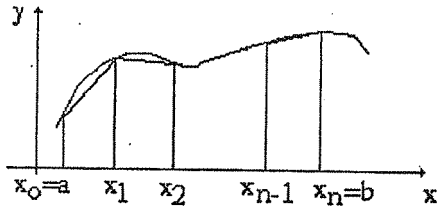


Fig. 17.2. Aproximarea integralei prin metoda trapezelor

Dacă integrala este aproximată prin aria subsecţiunilor unor curbe pătratice, iar intervalul [a,b] este divizat în 2n secţiuni egale:

$$h=\frac{b-a}{2n}=x_{j+1}-x_j>0 \qquad x_j=a+jh \qquad 1\le j\le 2n$$

integrala poate fi estimată prin expresia:

$$S_T=\frac{h}{3}\left[y_a+y_b+4(y_1+y_3+...+y_{2n-1})+2(y_2+y_4+...+y_{2n-2})\right]$$

cunoscută sub numele de formula generalizată a lui Simpson.

Îmbunătăţirea estimării integralei se poate face prin utilizarea unui pas mai fin.

17.1.2. Calculul numeric al integralelor pe intervale infinite

Pentru calculul integralelor de tipul:

$$S=\int_a^{\infty} f(x)dx$$

presupuse convergente, se folosesc două metode:

- schimbarea variabilei, care uneori poate transforma intervalul [a, +∞) într-un interval finit;
- separarea integralei în două integrale:

$$\int_a^{\infty} f(x)dx=\int_a^b f(x)dx+\int_b^{\infty} f(x)dx$$

În ultimul caz, prima integrală se evaluează prin una dintre metodele expuse anterior, iar pentru a doua se procedează astfel:

- fie se neglijează acest termen, dacă b este suficient de mare şi dacă :

$$\int_a^b f(x)dx >> \int_b^{2b} f(x)dx$$

- fie se aproximează forma asimptotică a lui f(x) printr-o funcţie g(x)≅f(x), care pentru

x suficient de mare permite calculul analitic al integralei funcţiei g(x);

17.1.3. Calculul numeric al integralelor funcţiilor cu singularităţi

Singularităţile sunt puncte ale unui interval mărginit pentru care funcţia este infinită. Un exemplu de integrală dintr-o funcţie cu singularităţi este:

$$S=\int_0^1 \frac{dx}{1-x}$$

Metodele de tratare a singularităţilor sunt:

- eliminarea acestora (dacă este posibil) prin schimbare de variabile, integrare prin părţi, etc;
- folosirea metodei lui Simpson, cu excluderea singularităţilor (înlocuirea limitei 1 cu 1-ε, cu ε suficient de mic în exemplul dat);
- folosirea cuadraturii bazate pe aproximarea funcţiei cu polinoame.

17.1.4. Funcţii MATLAB pentru integrarea numerică

Funcţiile MATLAB pentru integrare numerică sunt:

quad Calculează integrala prin metoda adaptiv-recursivă Simpson;
quad8 Calculează integrala prin metoda adaptiv-recursivă Newton Cotes de ordinul 8;
trapz Calculează integrala prin metoda trapezelor.

Funcţiile *quad* şi *quad8* sunt funcţii de integrare numerică (cuadratură) şi se apelează cu una dintre sintaxele:

quad('f ', a, b) *quad8*('f ', a, b)
quad('f ', a, b, tol) *quad8*('f ', a, b, tol)
quad('f ', a, b, tol, trace) *quad8*('f ', a, b, tol, trace)

unde:

f - este numele unui fişier funcţie care descrie funcţia f(x);
a, b - sunt limitele de integrare;
tol - este eroarea relativă admisă dintre doi paşi consecutivi;
trace - controlează afişarea pe ecran a valorilor intermediare.

Funcţiile *quad* şi *quad8* au acelaşi format de apelare.

Dacă funcţiile *quad* şi *quad8* sunt apelate cu parametrul "tol", integrarea funcţiei continuă să se rafineze până când eroarea relativă a estimării este mai mică decât toleranţa.

$$\frac{\text{Estim. anterioară} - \text{Estim. actuală}}{\text{Estim. actuală}} \le \text{toleranţa}$$

Dacă toleranţa este omisă, valoarea implicită este 10^{-3}. Dacă argumentul opţional "trace" este diferit de zero, se afişează valorile calculate în timpul procesului de calcul.

Funcţiile *quad* şi *quad8* pot trata unele singularităţi care se găsesc la unul dintre capetele intervalului de integrare, însă nu rezolvă singularităţile care se găsesc în interiorul intervalului.

Observaţie: Funcţiile *quad* şi *quad8* se utilizează dacă integrantul este exprimat sub forma unei funcţii analitice. Dacă funcţia de integrat este descrisă prin valori numerice, se aproximează mai întâi acestea printr-o funcţie corespunzătoare şi apoi aceasta va fi scrisă în sintaxa MATLAB, ca fişier M.

Funcţia *trapz* realizează calculul numeric al integralei prin metoda trapezelor. Se apelează cu una dintre sintaxele:

Z=*trapz*(X,Y) Z=*trapz*(Y)

Apelată cu două argumente, funcţia *trapz* calculează integrala funcţiei y(x). X este un vector coloană care conţine abscisele, iar Y este un vector coloană sau o matrice, cu acelaşi număr de linii ca X, care conţine valorile funcţiei corespunzătoare acelor abscise. Z este un scalar dacă Y este un vector sau un vector linie dacă Y este o matrice, în care sunt returnate ariile calculate pentru fiecare coloană a lui Y. Dacă funcţia *trapz* este apelată numai cu argumentul Y, distanţa dintre două abscise succesive este considerată unu. În acest caz limitele de integrare sunt a=1 şi b=N (N numărul de elemente ale vectorului Y).

Observaţie: Funcţia *trapz* presupune că integrantul este dat prin valori numerice în noduri echidistante ale intervalului de integrare. Dacă funcţia de integrat este descrisă sub forma analitică y=f(x), în primul rând se creează cei doi vectori care conţin valorile perechilor abscisă-ordonată (x-y). Acest lucru se realizează prin evaluarea funcţiei f(x) cu un pas adecvat, pentru x, şi apoi se aplică funcţia *trapz*.

17.2. Derivarea numerică

Derivata unei funcţii f(x) este viteza de variaţie a funcţiei în raport cu variabila x. Derivata este raportul dintre variaţia funcţiei f(x), notată cu df(x) şi variaţia lui x, notată cu dx:

$$f'(x)=\frac{df(x)}{dx}$$

Interpretarea geometrică a derivatei într-un punct este panta tangentei la graficul funcţiei în punctul considerat.

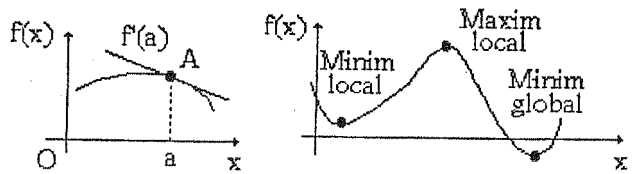


Fig. 17.3. Interpretarea geometrică a derivatei

Punctele cu derivata zero (puncte critice) pot reprezenta fie o regiune orizontală a funcţiei, fie un punct de maxim sau de minim local al funcţiei. Dacă derivata a doua în punctul critic este pozitivă, atunci valoarea funcţiei în punctul a este un minim local, iar dacă derivata a doua a punctului critic este negativă, atunci valoarea funcţiei în acel punct este un maxim local.

În contrast cu derivarea analitică derivarea numerică prezintă numeroase dificultăţi şi riscuri. De regulă, derivarea numerică amplifică micile erori, uneori îndepărtându-se semnificativ de la soluţia analitică.

17.2.1. Aproximarea derivatelor cu diferenţe finite

Aproximarea derivatei de ordinul I în punctul x_k, figura 17.4, poate fi făcută cu relaţia:

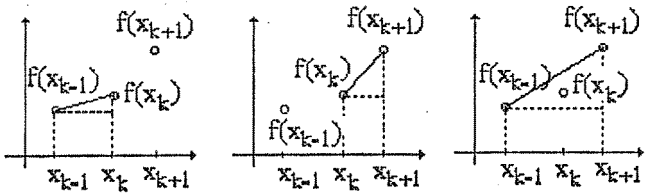


Fig. 17.4. Metode pentru calculul numeric al derivatei

$$f'(x) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

cunoscută sub denumirea de aproximare cu diferenţe regresive.
Aproximarea:

$$f'(x) \approx \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k}$$

este denumită aproximarea cu diferenţe progresive, iar:

$$f'(x) \approx \frac{f(x_{k+1}) - f(x_{k-1})}{x_{k+1} - x_{k-1}}$$

este o aproximare cu diferenţe centrale.

Precizia acestor estimări depinde de distanţa dintre punctele utilizate pentru calcul; creşterea preciziei se obţine prin micşorarea distanţei dintre punctele succesive.

17.2.2. Aproximarea numerică a derivatelor

Funcţia *diff* evaluează diferenţa elementelor succesive ale unui vector sau ale coloanei unei matrice; se apelează cu una dintre sintaxele:

Y= *diff*(X) Y= *diff*(X,n)

Dacă X este un vector linie sau coloană: X=[x(1), x(2), ... x(n)], funcţia *diff*(X) returnează vectorul diferenţelor elementelor adiacente, adică:

XD=[x(2)-x(1), x(3)-x(2), ... x(n)-x(n-1)]

Vectorul rezultat XD are cu un element mai puţin decât vectorul X.

Dacă X este o matrice, diferenţele se calculează pe fiecare coloană, prin scăderea elementului din linia imediat superioară, conform relaţiei:

diff(X) = X(2:m,:)- X(1:m-1,:)

unde m este numărul de linii.

Apelată cu argumentul n, funcţia returnează diferenţa de ordinul n a vectorului X. La

unele versiuni MATLAB această funcţie nu lucrează corect pentru diferenţe de ordin superior, recomandându-se scrierea unei funcţii utilizator (spre exemplu **diffn.m**), cu următorul conţinut:

```
function dY=diffn(X,n)
dY=X;
for i=1:n
      dY=diff(dY);
end
```

Funcţia *diff* permite aproximarea derivatei numerice a unei funcţii y(x) cu relaţia:

dy = *diff*(y) ./*diff*(x)

Spre exemplu pentru x=[0, 1, 2, 3, 4, 5] şi y=[2, 6, 3, 5, 9, 12]. Vectorul returnat de *diff*(x), este dx=[1 1 1 1 1], iar vectorul returnat cu *diff*(y) este dy=[4 -3 2 4 3]. Derivata dy/dx calculată cu expresia de mai sus este [4 -3 2 4 3].

Exemplul 17.2.1. Să se calculeze diferenţele de ordinul 1 şi 2 ale vectorului:

X = [1 4 5 9]

Cu secvenţa MATLAB:

```
X = [ 1  4  5  9 ]
Y = diff(X)
Z = diff(Y)
```

se obţin rezultatele:

Y = [3 1 4]
Z = [-2 3]

Exemplul 17.2.2. Să se calculeze diferenţele de ordinul 1 şi 2 ale matricei:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 9 & 8 & 7 \end{bmatrix}$$

Cu secvenţa MATLAB:

```
A=[1  2  3;4  5  6;9  8  7];
B = diff(A)
C=diff(B)
```

se obţine rezultatul:

```
B =                          C =
   3   3   3                    2   0  -2
   5   3   1
```

Exemplul 17.2.3. Să se calculeaze derivata funcţiei:

$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$

pe intervalul [-3, 4], utilizând aproximarea cu diferenţe progresive. Programul MATLAB care efectuează calculul acestei derivate, este :

```
x=-3:0.1:4
f=x.^5-3*x.^4-11.*x.^3+27*x.^2+10*x-24
df=diff(f)./diff(x);
xd=x(2:length (x));
plot(x,f,'g')
```

```
grid; hold on
plot(xd,df,'r')
```

Figura 17.5 conţine graficul funcţiei şi al derivatei acesteia. Zerourile derivatei corespund punctelor de maxim sau minim local ale acestei funcţii.

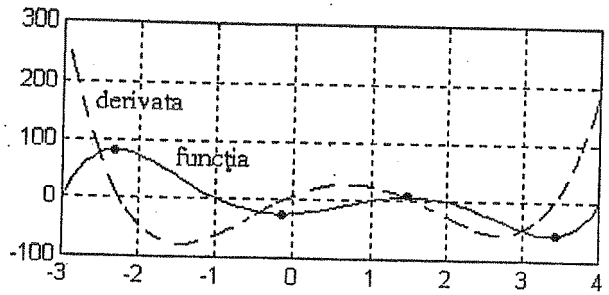


Fig. 17.5. Funcţia şi derivata ei, cu localizarea punctelor critice.

Pentru a afişa localizarea punctelor critice ale acestei funcţii, programul de mai sus trebuie completat cu următoarele instrucţiuni:

```
produs=df(1:length(df)-1).*df(2:length(df));
p_critic= xd(find(produs<0))
```

care returnează rezultatul:

p_critic = [-2.3000 -0.2000 1.5000 3.4000]

În ultima instrucţiune funcţia *find* determină indicii k ai elementelor la care derivata schimbă de semn. Aceste valori sunt utilizate cu vectorul xd pentru a afişa poziţiile la care sunt localizate punctele critice.

Pentru calculul derivatei cu diferenţe centrale se foloseşte următoarea secvenţă MATLAB:

```
x=-3:.1:4
f=x.^5-3*x.^4-11.*x.^3+27*x.^2+10*x-24;
df=f(3:length(f))-f(1:length(f)-2);
dx=x(3:length(x))-x(1:length(x)-2);
dy=df./dx;
xd=x(2:length(x)-1)
pr=df(1:length(df)-1).*df(2:length(df));
p_cr= xd(find(pr<0)+1)
f_cr=f(find(pr<0)+1)
plot(x,f,'g')
hold on
plot(xd,df,'r', p_cr, f_cr,'ob')
grid; hold off
```

În exemplul analizat s-a presupus că se cunoaşte expresia analitică a funcţiei de derivat. În multe aplicaţii însă, funcţia care trebuie derivată constă dintr-o succesiune de date. În acest caz, se recomandă interpolarea datelor utilizând metodele prezentate anterior.

17.2.3. Derivarea numerică bazată pe polinomul de interpolare Newton

Se presupun cunoscute valorile $y_i=f(x_i)$ în nodurile de interpolare $x_0, x_1, \ldots, x_n$, echidistante. Dacă se consideră notaţiile:

$h = x_{i+1} - x_i$ - pasul de divizare;

$\Delta y_i = f(x_{i+1}) - f(x_i)$ - diferenţele de ordinul 1;

$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$ - diferenţele de ordinul 2;

$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i$ - diferenţele de ordinul 3;

relaţiile pentru calculul primelor două derivate, luând în considerare diferenţele finite numai până la ordinul 4, sunt:

$$f'(x_i) = \frac{1}{h}\left[\Delta y_i - \frac{\Delta^2 y_i}{2} + \frac{\Delta^3 y_i}{3} - \frac{\Delta^4 y_i}{4}\right]$$

$$f''(x_i) = \frac{1}{h^2}\left[\Delta^2 y_i - \Delta^3 y_i + \frac{11}{12}\Delta^4 y_i\right]$$

Utilizând funcţia *diff*, aceste relaţii se implementează cu uşurinţă.

Exemplul 17.2.4. Scrieţi un program care să calculeze primele două derivate ale funcţiei lg(x), în punctul x=50.

Cu secvenţa MATLAB:

```
x=50:2:58;
h=x(2)-x(1);
y=log10(x);
d1y=diff(y);
d2y=diff(d1y);
d3y=diff(d2y);
d4y=diff(d3y);
d1f=(1/h)*(d1y(1)-d2y(1)/2+d3y(1)/3-d4y(1)/4)
d2f=(1/h^2)*(d2y(1)-d3y(1)+(11/12)*d4y(1))
```

se obţin rezultatele:

d1f = 0.0087

d2f = -1.7355e-004

17.2.4. Aproximarea numerică a Laplaceanului

Funcţia MATLAB *del2* calculează Laplaceanul discret în cinci puncte. Se apelează cu sintaxa:

V=*del2*(U)

în care elementele matricei V, de aceeaşi dimensiune cu matricea U, sunt calculate cu relaţia:

$$v_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{4} - u_{i,j}$$

În cazul nodurilor care aparţin marginilor sau colţurilor (care au numai trei şi respectiv două noduri vecine), media este calculată numai cu aceste noduri.

Dacă matricea U este asociată funcţiei u(x,y) evaluată în nodurile unei reţele cu pas constant, atunci:

D=4*del2(U)

este aproximarea cu diferenţe finite a operatorului diferenţial Laplace aplicat lui u, adică:

$$\Delta^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

Exemplul 17.2.5. Calculaţi aproximarea numerică a Laplaceanului funcţiei $u(x,y)=3x^2+2y^3$.

Cu secvenţa MATLAB:

```
[x, y]=meshgrid(-4:2:4, -2:2);
U=3*x.^2+2*y.^3;
V=4*del2(U)
```

se obţin rezultatele:

V =

12	48	48	48	12
-24	12	12	12	-24
0	24	24	24	0
24	36	36	36	24
36	48	48	48	36

17.2.5. Aproximarea numerică a gradientului

Funcţiile MATLAB:

gradient Aproximează gradientul unei funcţii;
quiver Reprezintă grafic orientarea unui câmp de vectori

Funcţia *gradient* aproximează numeric gradientul unei funcţii; se apelează cu sintaxa:

[PX,PY]= *gradient*(Z,DX,DY)

care returnează derivatele parţiale numerice ale matricei Z în matricele PX = dZ/dx şi PY = dZ/dy. DX şi DY pot fi scalari care conţin pasul de eşantionare pe direcţiile X şi Y sau pot fi vectori care conţin toate coordonatele după aceste direcţii. Dacă DX şi DY sunt omişi, se consideră că DX = DY = 1.

Dacă funcţia gradient este apelată cu una dintre sintaxele:

dYX=*gradient*(Y) dYX=*gradient*(Y,DY)

atunci ea returnează un vector care conţine derivata numerică dY/dX.

Funcţia *quiver* reprezintă grafic mici săgeţi (vectori orientaţi) la fiecare pereche (X,Y) a matricelor X şi Y; se apelează cu una dintre sintaxele:

quiver(X, Y, DX, DY) *quiver*(DX, DY)
quiver(X, Y, DX, DY, S)

Perechile (DX, DY) din matricele DX şi DY determină direcţia şi modulul câmpului de vectori orientaţi. Dacă X şi Y sunt vectori, X corespunde coloanelor lui DX şi DY, iar Y corespunde liniilor acestora (X trebuie să aibă lungimea egală cu numărul de coloane, iar Y cu numărul de linii).

Dacă X şi Y sunt omişi, se consideră X=1:n şi Y=1:m, unde m este numărul de linii

al matricelor DX (DY), iar n numărul de coloane al acestora.

Argumentul S, care este un scalar, permite aplicarea unui factor de scală lungimii vectorilor orientaţi de pe reprezentarea grafică. Se pot folosi toate tipurile de linii sau culori, specificate la instrucţiunea *plot*.

Exemplul 17.2.6. Să se reprezinte grafic gradientul unui câmp de vectori asociat funcţiei:

$$z(x,y) = x\,e^{-x^2-y^2}$$

în domeniul XxY=[-2, 2]x[-2, 2].

Cu secvenţa MATLAB:

```
xp = -2:.2:2;
yp = -2:.2:2;
[x,y] = meshgrid(xp,yp);
z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z,.2,.2);
contour(x,y,z),hold on,
quiver(x,y,px,py,1.5,'g'), hold off
```

se obţine reprezentarea grafică din figura 17.6.

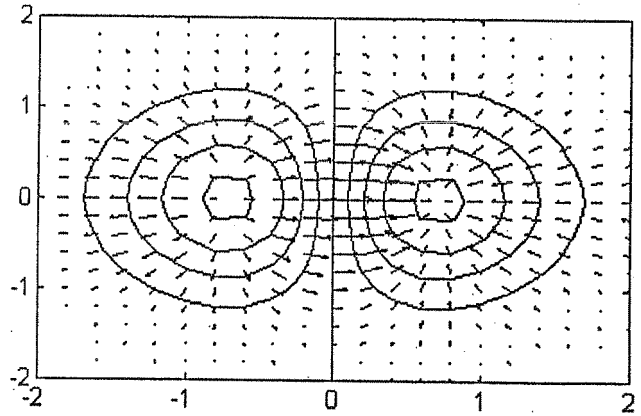


Fig. 17.6. Reprezentarea grafică a gradientului unui câmp de vectori.

18. INTEGRAREA NUMERICĂ A ECUAŢIILOR DIFERENŢIALE

În acest capitol se prezintă funcţiile MATLAB care implementează metodele Runge-Kutta pentru rezolvarea ecuaţiilor diferenţiale de ordinul întâi şi metodele de rezolvare a ecuaţiilor diferenţiale de ordin superior, prin convertire în sisteme de ecuaţii diferenţiale de ordinul întâi.

18.1. Ecuaţii diferenţiale de ordinul întâi

O ecuaţie diferenţială de ordinul întâi are forma:

$$y' = \frac{dy}{dx} = g(x,y)$$

unde x este variabila independentă, iar y este funcţia necunoscută. Următoarele exemple sunt ecuaţii diferenţiale de primul ordin.

1. $y'=g_1(x,y)=3x^2$
2. $y'=g_2(x,y)=-0.1y$
3. $y'=g_3(x,y)=3y+e^{2x}$

Cu condiţiile iniţiale respectiv: y(0)=-7.5, y(0)=4, y(0)=3, aceste ecuaţii au soluţiile:

1. $y=x^3-7.5$
2. $y=4\,e^{-0.1x}$
3. $y=4\,e^{3x}-e^{2x}$

Deşi soluţiile analitice ale ecuaţiilor diferenţiale sunt preferate, pentru multe ecuaţii astfel de soluţii nu pot fi obţinute.

Metodele numerice cele mai cunoscute pentru rezolvarea ecuaţiilor diferenţiale sunt metoda Euler şi metoda Runge-Kutta. Ambele aproximează funcţiile utilizând descompunerea în serii Taylor:

$$f(b) = f(a) + (b-a)f'(a) + \frac{(b-a)^2}{2!}f''(a) + \ldots + \frac{(b-a)^n}{n!}f^{(n)}(a) + \ldots$$

Aproximarea de ordinul întâi foloseşte funcţia şi derivata de ordinul unu a acestela:

$$f(b) \approx f(a) + (b-a)f'(a)$$

iar aproximarea de ordinul doi utilizează funcţia şi derivatele de ordinul unu şi doi:

$$f(b) \approx f(a) + (b-a)f'(a) + \frac{(b-a)^2}{2!}f''(a)$$

Creşterea numărului de termeni ai seriei Tayor măreşte precizia aproximării.

18.2. Funcţiile MATLAB pentru integrarea numerică a ecuaţiilor diferenţiale

Funcţiile MATLAB pentru integrarea numerică a ecuaţiilor diferenţiale de ordinul întâi sunt:

ode23 Rezolvă ecuaţii diferenţiale prin metoda Runge-Kutta de ordinul 2/3;
ode45 Rezolvă ecuaţii diferenţiale prin metoda Runge-Kutta de ordinul 4/5.

Funcţiile *ode23* şi *ode45* se apelează cu una dintre sintaxele:

```
[x,y]=ode23('yprim',x0,xf,y0)
[x,y]=ode45('yprim',x0,xf,y0)
[x,y]=ode23('yprim',x0,xf,y0,tol,trace)
[x,y]=ode45('yprim',x0,xf,y0,tol,trace)
```

unde:

- yprim - este o variabilă şir cu numele unui fişier-M care defineşte derivata funcţiei necunoscute y;
- x0 - este valoarea iniţială a variabilei x;
- xf - este valoarea finală a variabilei x;
- y0 - este un vector coloană conţinând condiţiile iniţiale;
- tol - este precizia dorită a soluţiei (opţională). Implicit tol=10^{-3} pentru *ode23* şi tol=10^{-6} pentru *ode45*;
- trace - este un parametru care asigură tipărirea rezultatelor intermediare opţional). Valoarea implicită este zero (fără tipărire).

Numărul de puncte x_i aparţinând intervalului (x0, xf), în care funcţiile *ode23* şi *ode45* returnează valorile y_i ale necunoscutei, este un parametru intern al acestora. Funcţiile *ode23* şi *ode45* apelează la metode de integrare Runge-Kutta-Fehlberg, cu stabilirea automată a pasului. Algoritmul Runge-Kutta cu pas automat îşi alege pasul în funcţie de gradientul soluţiei, respectiv un pas mai mare atunci când gradientul y'(x) are o valoare mai mică.

Funcţiile MATLAB au două argumente de ieşire; valorile variabilei x şi mărimile corespunzătoare ale necunoscutei y, care reprezintă soluţia numerică a ecuaţiei diferenţiale.

Pentru a ilustra modul de utilizare al funcţiei *ode23*, se prezintă calculul numeric al soluţiilor ecuaţiilor diferenţiale date ca exemplu la începutul paragrafului 18.1. Soluţia numerică (x, yn) este reprezentată cu markere, iar soluţia analitică (x, ya) este reprezentată cu linie continuă.

Exemplul 18.2.1. Să se integreze ecuaţia diferenţială $y'=3x^2$ pe intervalul [2,4], cu condiţia iniţială y(2)=0.5.

Se creează un fişier funcţie:

```
function dy=g1(x,y)
dy=3*x^2;
```

cu numele **g1.m**, care se apelează cu secvenţa MATLAB:

```
[x, yn]=ode23('g1',2, 4, 0.5);
```

```
ya =x.^3-7.5;
plot(x, yn,'o', x, ya); grid
```

obţinându-se soluţia numerică reprezentată comparativ cu cea analitică în figura 18.1.

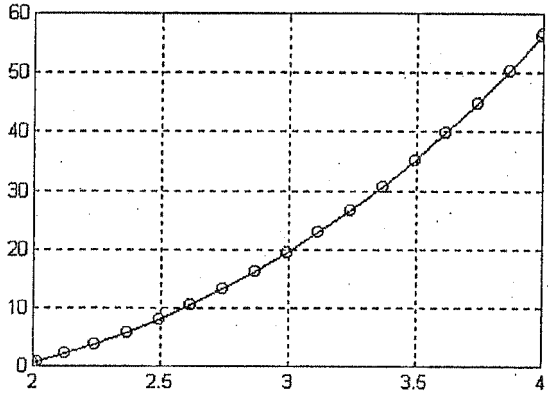


Fig. 18.1. Soluţiile numerică şi analitică ale ecuaţiei $y' = 3x^2$, cu y(0)=0.5.

Exemplul 18.2.2. Să se integreze ecuaţia diferenţială y'=-0.1y pe intervalul [0, 5], cu condiţia iniţială y(0)=4.
Se creează un fişier funcţie:

```
function dy =g2(x,y)
dy=-0.1*y;
```

cu numele **g2.m**, care se apelează cu secvenţa MATLAB:

```
[x, yn]=ode23('g2',0, 5,4);
ya=4.*exp(-0.1.*x);
plot(x, yn,'o', x, ya); grid
```

obţinându-se soluţia numerică reprezentată comparativ cu cea analitică în figura 18.2.

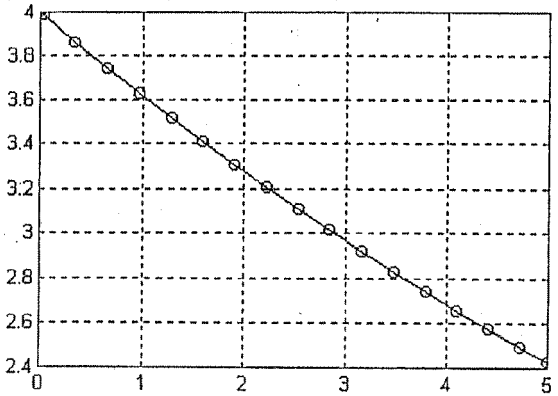


Fig. 18.2. Soluţiile numerică şi analitică ale ecuaţiei $y' = -0.1y$, cu y(0)=4.

Exemplul 18.2.3. Să se integreze ecuaţia diferenţială $y' = 3y+e^{2x}$ pe intervalul [0, 2], cu condiţia iniţială y(0)=3.
Se creează un fişier funcţie:

```
function dy=g3(x,y)
dy=3*y+exp(2*x);
```

cu numele **g3.m**, care se apelează cu secvenţa MATLAB:

```
[x,yn]=ode23('g3', 0,2, 3);
ya=4*exp(3 .*x)-exp(2 .*x);
plot(x, yn,'o', x, ya); grid
```

obţinându-se soluţia numerică reprezentată comparativ cu cea analitică în figura 18.3.

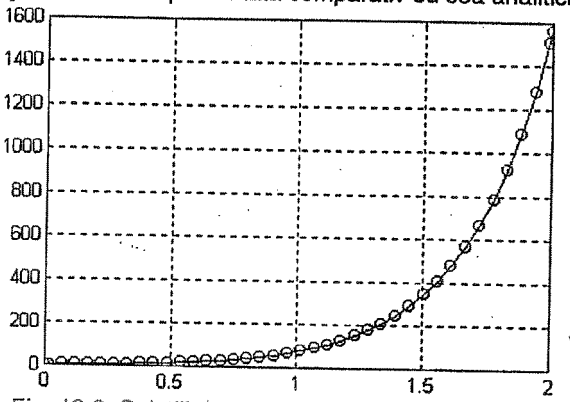


Fig. 18.3. Soluţiile numerică şi analitică ale ecuaţiei $y' = 3y+e^{2x}$, cu y(0)=3.

18.3. Integrarea ecuaţiilor diferenţiale de ordin superior şi a sistemelor de ecuaţii diferenţiale

O ecuaţie diferenţială de ordin superior este echivalentă cu un sistem de ecuaţii diferenţiale de ordinul întâi cuplate. Spre exemplu, ecuaţia diferenţială de ordinul n :

$$y^{(n)} = g\left(x, y, y', y'', \ldots, y^{(n-1)}\right)$$

la care se asociază notaţiile:

$$\begin{cases} y_1(x) = y^{(n-1)} \\ y_2(x) = y^{(n-2)} \\ \ldots \\ y_{n-2}(x) = y'' \\ y_{n-1}(x) = y' \\ y_n(x) = y \end{cases}$$

este echivalentă cu următorul sistem de n ecuaţii diferenţiale de ordinul întâi:

$$\begin{cases} y_1' = y^{(n)} = g(x, u_n, u_{n-1}, \ldots, u_1) \\ y_2' = y_1 \\ \ldots \\ y_{n-2}' = y_{n-3} \\ y_{n-1}' = y_{n-2} \end{cases}$$

Exemplul 18.3.1. Fie ecuaţia diferenţială liniară de ordinul doi:

$$y'' + (y^2 - 1)y' + y = 0$$

cu condiţiile iniţiale: $y(0) = 0.25$, $y'(0) = 0$.

cunoscută sub numele de ecuaţia Van der Pol. Această ecuaţie diferenţială este echivalentă cu sistemul:

$$\begin{cases} y_1' = y_1(1 - y_2^2) - y_2 \\ y_2' = y_1 \end{cases}$$

Prima etapă în rezolvarea acestui sistem este crearea unei funcţii-M, conţinând cele două ecuaţii diferenţiale; fie acesta **ecdif.m**:

```
function yprim=ecdif(x,y)
yprim=zeros(2,1);
yprim(1)=y(1).*(1-y(2).^2)-y(2);
xprim(2)=y(1);
```

Pentru integrarea sistemului de ecuaţii diferenţiale de ordinul întâi, pe intervalul [0, 20], se utilizează instrucţiunile MATLAB:

```
t0=0;tf=20;
x0=[0 0.25]; % Conditii initiale
[t,x]=ode23('ecdif',t0,tf,x0);
plot(t,x(:,1), '--g',t,x(:,2), '-r');grid
```

Se obţin graficele reprezentate în figura 18.4.

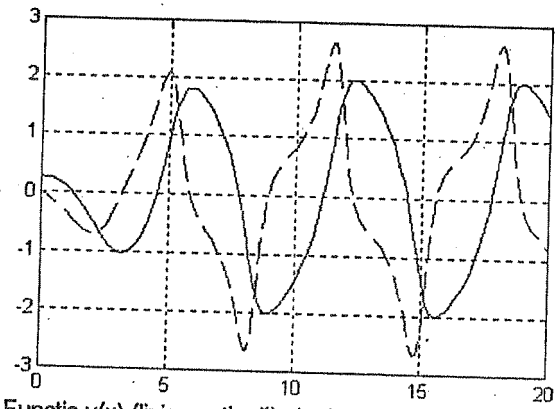


Fig. 18.4. Funcţia y(x) (linia continuă) şi prima derivată y′ (x) (linia întreruptă)

- soluţii ale ecuaţiei diferenţiale $y'' + (y^2-1)y' + y = 0$.

Exemplul 18.3.2. Să se rezolve ecuaţia diferenţială:

$$y^{(5)} = 0$$

pe intervalul [0, 2], cu condiţiile iniţiale:

$$y(0) = 1, \quad y'(0) = 0, \quad y''(0) = -1, \quad y'''(0) = 0, \quad y^{(4)}(0) = 1$$

Soluţia analitică este:

$$y = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 \quad ; \quad x \in R$$

Pentru rezolvarea acestei ecuaţii diferenţiale se scriu două fişiere:

- unul care să descrie sistemul de n ecuaţii diferenţiale de ordinul unu, echivalent cu ecuaţia diferenţială de ordinul n ;
- altul care să precizeze limitele de integrare şi condiţiile iniţiale.

Ecuaţia diferenţială de ordinul 5 este echivalentă cu sistemul de 5 ecuaţii diferenţiale de ordinul unu:

$$\begin{cases} y_1' = 0 & \text{derivata a 5-a} \\ y_2' = y_1 & \text{derivata a 4-a} \\ y_3' = y_2 & \text{derivata a 3-a} \\ y_4' = y_3 & \text{derivata a 2-a} \\ y_5' = y_4 & \text{prima derivată} \end{cases}$$

Vectorul condiţiilor iniţiale, yp, conţine valorile derivatelor, în ordine descrescătoare. Fişierul care descrie sistemul de ecuaţii diferenţiale, **ec_dif1.m**, este următorul:

```
function yp = ec_dif1(x,y)
yp=zeros(5,1);% vectorul derivatelor
yp(1)=0; %derivata de ordin 5
yp(2)=y(1); %derivata de ordin 4
yp(3)=y(2); %derivata de ordin 3
yp(4)=y(3); %derivata de ordin 2
yp(5)=y(4); %derivata de ordin 1
```

Vectorul y0, numit vector al condiţiilor iniţiale, este un vector coloană care conţine valorile iniţiale ale necunoscutei şi primelor n-1 derivate ale acesteia (dacă ecuaţia este de ordinul n); în cazul de faţă;

$$y0 = [\, y^{(4)}(0) \;\; y'''(0) \;\; y''(0) \;\; y'(0) \;\; y(0) \,]'$$

Limitele de integrare şi vectorul condiţiilor iniţiale se găsesc în fişierul programului principal, **testdif1.m**, al cărui conţinut este:

```
x0 = 0;  xf = 2;
y0 = [ 1;  0;  -1;  0;  1 ];
[x,y] = ode23('ec_dif1', x0, xf, y0);
z = 1 - (1/2).*x.^2 + (1/24).*x.^4;
plot(x,y(:,5),'o', x,z); grid
```

Se obţin graficele reprezentate în figura 18.5.

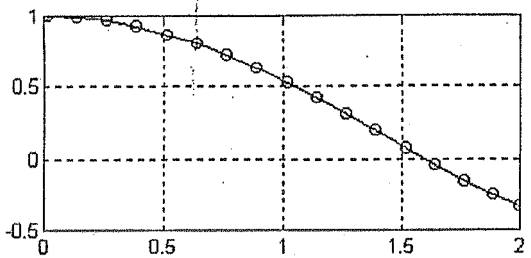


Fig. 18.5. Reprezentarea grafică a soluţiilor analitică şi numerică a ecuaţiei diferenţiale $y^{(5)} = 0$ pe intervalul [0, 2].

Exemplul 18.3.3. Să se integreze ecuaţia diferenţială:

$$y'' + y = 0$$

pe intervalul [0, 2π], cu condiţiile iniţiale:

$$y(0) = 0, \quad y'(0) =$$

Soluţia analitică este:

$$y = \sin(x)$$

Se creează fişierul funcţie **ec_dif2.m**, care descrie sistemul de ecuaţii diferenţiale, cu următorul conţinut:

```
function yp = ec_dif2(x,y)
yp=zeros(2,1);% vectorul derivatelor
ydm=-y(2); % derivata de ordin 2
yp(1)=ydm; %derivata de ordin 2
yp(2)=y(1); %derivata de ordin 1
```

şi apoi fişierul programului principal, **testdif2.m**, care apelează fişierul funcţie **ec_dif2.m**, şi care are conţinutul:

```
x0=0;  xf=2*pi; %limitele de integrare
y0=[1; 0]; %vectorul cond. initiale
[x,y]=ode23('ec_dif2', x0, xf, y0);
z = sin(x);
plot(x,y,x,z,'o'); grid
```

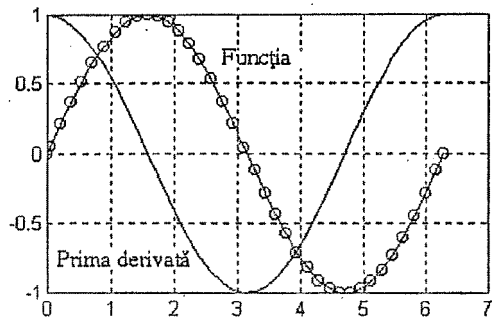


Fig. 18.6. Funcţia şi prima derivată a ecuaţiei diferenţiele $y'' + y = 0$.

Exemplul 18.3.4. Să se rezolve ecuaţia diferenţială:

$$y^{(4)} - 3y'' - 4y = 0$$

cu condiţiile iniţiale:

$$y(0) = 0, \quad y'(0) = 0, \quad y''(0) = 0, \quad y'''(0) = 2$$

şi soluţia analitică:

$$y = -\frac{2}{5}\sin(x) - \frac{1}{10}e^{-2x} + \frac{1}{10}e^{2x} \quad ; \quad x \in R$$

Pentru rezolvarea acestei ecuaţii cu MATLAB-ul, se exprimă derivata de ordinul cel mai mare (4 în acest caz) ca funcţie de celelalte derivate şi variabila independentă:

$$y^{(4)} = 3y'' + 4y$$

Cu notaţiile:

$$\begin{cases} y^{(4)} = 3y(2) + 4y(4) & \text{derivata a 4 - a} \\ y''' = y(1) & \text{derivata a 3 - a} \\ y'' = y(2) & \text{derivata a 2 - a} \\ y' = y(3) & \text{prima derivată} \end{cases}$$

se creează fişierul funcţie, fie spre exemplu **ec_dif3.m**, care descrie sistemul de ecuaţii diferenţiale, cu următorul conţinut:

```
function yp = ec_dif3(x,y)
yp=zeros(4,1);% vectorul derivatelor
ydm=3*y(2)+4*y(4); %derivata de ordin 4
yp(1)=ydm; %derivata de ordin 4
yp(2)=y(1); %derivata de ordin 3
yp(3)=y(2); %derivata de ordin 2
yp(4)=y(3); %derivata de ordin 1
```

şi apoi fişierul programului principal, **testdif3.m**, în care se găsesc limitele de integrare (x0 şi xf) şi vectorul coloană, y0, al condiţiilor iniţiale:

$$y0 = [\ y'''(0)\ \ y''(0)\ \ y'(0)\ \ y(0)\ ]'$$

Conţinutul programului principal este:

```
x0=0; xf=2.5; tol=10^(-9);
y0=[2; 0; 0; 0];
[x,y]=ode45('ec_dif3',x0,xf,y0,tol,1);
z=-(2/5).*sin(x)-...
(1/10).*exp(-2.*x)+(1/10).*exp(2.*x);
plot(x,y(:,4), x,z,'o'); grid
```

Acest program utilizează pentru integrare funcţia *ode45*, cu toleranţa 10^{-9} şi afişarea valorilor intermediare.

19. FUNCŢII SPECIALE

Funcţiile speciale prezentate în acest capitol sunt: funcţiile Bessel de prima şi a doua speţă, funcţia Beta, funcţia eroare, funcţiile eliptice şi funcţia Gamma. Astfel:

besselj	Calculează funcţiile Bessel de speţa întâi;
bessely	Calculează funcţiile Bessel de speţa a doua;
besseli	Calculează funcţiile Bessel modificate de speţa întâi;
besselk	Calculează funcţiile Bessel modificate de speţa a doua;
beta	Calculează funcţia Beta;
betainc	Calculează funcţia Beta incompletă;
betaln	Calculează funcţia Beta logaritmică;
erf	Calculează funcţia eroare;
erfc	Calculează funcţia eroare complementară;
erfinv	Calculează inversa funcţiei eroare;
gamma	Calculează funcţia Gamma;
gammainc	Calculează funcţia Gamma incompletă;
gammaln	Calculează funcţia Gamma logaritmică.

19.1. Ecuaţia şi funcţiile Bessel

Ecuaţia diferenţială :

$$x^2y''(x)+xy'(x)+(x^2-\nu^2)y(x)=0 \quad , \quad \text{pentru} \quad x>0 \ \text{ si } \nu=\text{const.}\geq 0$$

este o ecuaţie Bessel. Această ecuaţie apare în rezolvarea unor probleme de câmp cu simetrie cilindrică.

Soluţia generală a ecuaţiei diferenţiale Bessel pentru $\nu\neq$întreg este:

$$y=C_1J_\nu(x)+C_2J_{-\nu}(x).$$

unde C_1 şi C_2 sunt constante reale, este funcţia Bessel de speţa întâi şi ordin ν.

Soluţia generală a ecuaţiei Bessel pentru ν=n=întreg, este:

$$y=C_1J_n(x)+C_2Y_n(x)$$

unde C_1 şi C_2 sunt constante reale, iar Y_n este funcţia Bessel de speţa a doua şi ordin n.

Ecuaţia diferenţială :

$$x^2y''(x)+xy'(x)-(x^2+\nu^2)y(x)=0 \quad , \quad \text{pentru} \quad x>0 \ \text{ si } \nu=\text{const.}\geq 0$$

este cunoscută sub numele de ecuaţie Bessel modificată.

Soluţia generală a ecuaţiei diferenţiale Bessel modificate pentru $\nu\neq$întreg este:

$$y=C_1I_\nu(x)+C_2I_{-\nu}(x)$$

unde C_1 şi C_2 sunt constante reale, iar I_ν este funcţia Bessel modificată, de speţa întâi şi ordin ν.

Soluţia generală a ecuaţiei diferenţiale Bessel modificate, pentru ν=n=întreg, este:

$$y=C_1I_n(x)+C_2K_n(x)$$

unde C_1 şi C_2 sunt constante reale, iar I_n şi K_n sunt funcţiile Bessel modificate de speţa unu şi doi şi ordin n.

Funcţiile Bessel prezentate mai sus, se calculează în MATLAB după cum

urmează:

J=*besselj*(niu, X) - calculează funcţiile Bessel de speţa întâi, pentru ordinul „niu" real pozitiv şi argumentul X. X şi niu pot fi vectori de dimensiunile m şi respectiv n, caz în care J va fi o matrice cu dimensiunea m x n. Elementul din poziţia (k,j) este J(k,j)=*besselj*(niu(j),X(k)). Dacă argumentul „niu" este vector, el trebuie să aibă incrementul 1 şi să aparţină intervalului [0, 1000].

- Exemple de utilizare a funcţiei:

```
J=besselj(3:9,(10:.2:20)')
J=besselj(1/3:1:25/3,(-20:.5:30))
```

N=*bessely*(niu, X) - calculează funcţiile Bessel de speţa a doua, pentru ordinul „niu" real pozitiv şi argumentul X. X şi niu pot fi vectori de dimensiunile m şi respectiv n, caz în care N va fi o matrice cu dimensiunea m x n. Elementul din poziţia (k,j) este N(k,j)=*bessely*(niu(j),X(k)). Dacă argumentul „niu" este vector, el trebuie să aibă incrementul 1 şi să aparţină intervalului [0, 1000].

- Exemplu de utilizare a funcţiei:

```
N=bessely(3:9,(10:.2:20)')
```

I=*besseli*(niu, X) - calculează funcţiile Bessel modificate de speţa întâi, pentru ordinul „niu" real pozitiv şi argumentul X pozitiv. X şi „niu" pot fi vectori de dimensiunile m şi respectiv n, caz în care I va fi o matrice cu dimensiunea m x n, Elementul din poziţia (k,j) este I(k,j)= *besseli*(niu(j),X(k)). Argumentul niu trebuie să aibă incrementul 1 şi să aparţină intervalului [0, 1000].

- Exemple de utilizare a funcţiei:

```
I=besseli(3:9,(10:.2:20)')
I=besseli(1/3:1:25/3,(-20:.5:30))
```

K=*besselk*(niu, X) - calculează funcţiile Bessel modificate de speţa a doua, pentru ordinul „niu" real pozitiv şi argumentul X pozitiv. X şi „niu" pot fi vectori de dimensiunile m şi respectiv n, caz în care I va fi o matrice cu dimensiunea m x n. Elementul din poziţia (k,j) este I(k,j)= *besseli*(niu(j),X(k)). Argumentul „niu" trebuie să aibă incrementul 1 şi să aparţină intervalului [0, 1000].

- Exemplu de utilizare a funcţiei:

```
K=besselk(3:9,(10:.2:20)')
```

F=*bessela*(niu,X) - calculează funcţiile Bessel pentru argument complex.

F=*bessel*(niu,X) - apelează *besselj* dacă X este real de speţa întâi, *besseli* dacă X este imaginar şi *bessela* dacă X este complex;

Exemplul 19.1.1. Să se reprezinte grafic funcţiile Bessel de speţa întâi şi de ordine 0, 1, 2.

```
x = 0:.25:20;
niu=[0  1  2];
J=besselj(niu,x);
plot(x,J); grid
```

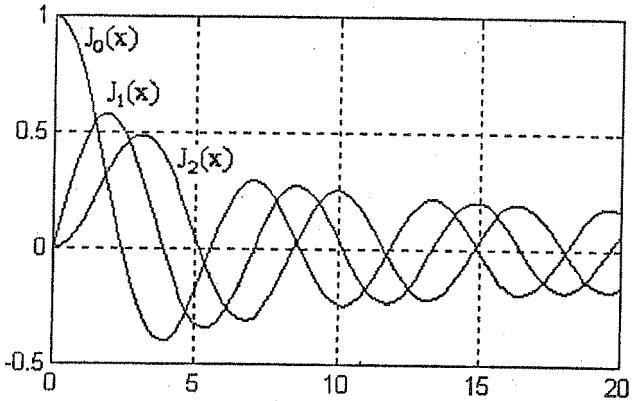


Fig. 19.1. Funcţiile Bessel de speţa întâi şi ordin întreg.

Exemplul 19.1.2. Să se reprezinte grafic funcţiile Bessel modificate de speţa întâi şi de ordine 0 , 1 şi 2.

```
x = 0:.1:2;
niu=[0  1  2];
N=besseli(niu,x);
plot(x,N); grid
```

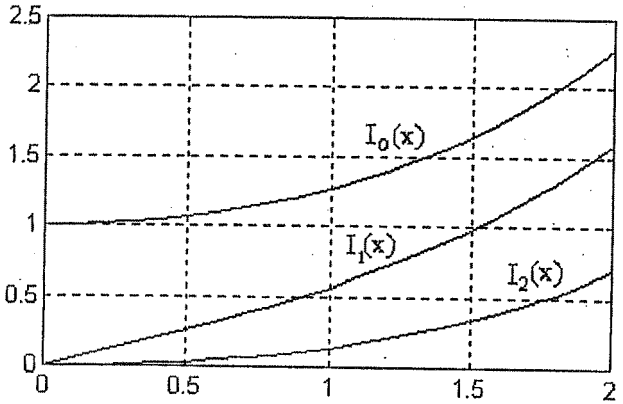


Fig. 19.2. Funcţiile Bessel modificate de speţa întâi şi ordin întreg.

19.2. Funcţia Gamma

Funcţia Gamma este definită de integrala:

$$\Gamma(a)=\int_0^{\infty} t^{a-1}e^{-t}dt \quad , \quad a\in\Re$$

Dacă argumentul a este întreg, funcţia Gamma este identică cu funcţia factorial:

$$\Gamma(n+1)=n!$$

Această funcţie satisface relaţia de recurenţă:

$$\Gamma(a+1)=a\Gamma(a)$$

Dacă funcţia Gamma este cunoscută pentru a>1, valorile acesteia pentru a<1 se pot obţine cu relaţia:

$$\Gamma(1-a)=\frac{\pi}{\Gamma(a)\sin(\pi a)}=\frac{\pi a}{\Gamma(1+a)\sin(\pi a)}$$

Funcţia MATLAB care calculează funcţia Γ(a) se apelează cu sintaxa:

G=*gamma*(a)

Funcţia Gamma incompletă este definită de relaţia:

$$P(x,a)=\frac{1}{\Gamma(a)}\int_0^{x} t^{a-1}e^{-t}dt \quad , \quad x\in\Re,\ a\in\Re_+$$

şi are limitele:

$$P(0,a)=0 \quad \text{şi} \quad P(\infty,a)=$$

Funcţia P(x,a) se apelează cu sintaxa:

P=*gammainc*(x,a)

Argumentul x trebuie să fie real (scalar sau vector), iar a trebuie să fie scalar real şi pozitiv.

Funcţia MATLAB *gammaln* returnează logaritmul natural din valorile lui *gamma*; se apelează cu sintaxa:

Gl=*gammaln*(a)

19.3. Funcţia Beta

Funcţia Beta este descrisă de expresia:

$$B(z,w)=B(w,z)=\int_0^{1} t^{z-1}(1-t)^{w-1}dt=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$$

Se apelează cu sintaxa:

B=*beta*(z,w)

Funcţia Beta incompletă descrisă de expresia:

$$I_x(z,w)=\frac{1}{B(z,w)}\int_0^{x} t^{z-1}(1-t)^{w-1}dt$$

are limitele:

$$I_0(z,w)=0 \quad \text{si} \quad I_1(z,w)=$$

şi satisface relaţia de simetrie:

$$I_x(z,w)=1-I_{1-x}(w,z)$$

Funcţia $I_x(z,w)$ se apelează cu sintaxa:

I=*betainc*(x,z,w)

Funcţia MATLAB *betaln* returnează logaritmul natural din valorile lui *beta*; se apelează cu sintaxa:

BI=*betaln*(z,w)

Variabilele z şi w pot fi vectori sau matrice de aceeaşi dimensiune.

19.4. Funcţia eroare

Funcţia eroare *erf* şi complementara funcţiei eroare sunt cazuri speciale ale funcţiei Gamma incomplete; sunt definite prin relaţiile:

$$\mathrm{erf}(x)=\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

şi:

$$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt=1-\mathrm{erf}(x)$$

Aceste funcţii au următoarele limite şi respectă relaţiile de simetrie:

$$\mathrm{erf}(0)=0\,;\quad \mathrm{erf}(\infty)=1\,;\quad \mathrm{erf}(-x)=-\mathrm{erf}(x)\,;$$

$$\mathrm{erfc}(0)=1\,;\quad \mathrm{erfc}(\infty)=0\,;\quad \mathrm{erfc}(-x)=2-\mathrm{erfc}(x)$$

Relaţia dintre funcţia eroare şi funcţia Gamma incompletă este:

$$\mathrm{erf}(x)=P(x^2,0.5)\ ,\quad x\ge 0$$

Funcţiile *erf* şi *erfc* se apelează cu sintaxele:

y = *erf*(x) y=*erfc*(x)

Inversa funcţiei eroare, *erfinv*, returnează valoarea lui x pentru valori cunoscute ale lui y; se apelează cu sintaxa:

x = *erfinv*(y)

Valorile lui y trebuie să fie în intervalul [-1 , 1]; x rezultă în intervalul (-∞, +∞).

Pentru a calcula valoarea funcţiei eroare generalizate:

$$\mathrm{erf}(x,y)=\frac{2}{\sqrt{\pi}}\int_x^y e^{-t^2}dt=\mathrm{erf}(y)-\mathrm{erf}(x)$$

se utilizează secvenţa:

E=*erf*(y)-*erf*(x)

Argumentul x poate fi scalar, vector sau matrice, rezultatul fiind de aceleaşi dimensiuni.

19.5. Funcţia de repartiţie Laplace

Funcţia densităţii de probabilitate a repartiţiei normale normate este definită de relaţia:

$$f(z)=\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}$$

Calculul acesteia în MATLAB se face cu secvenţa:

```
f=1./(sqrt(2*pi))*exp(-z.^2/2)
```

unde z poate fi vector sau scalar. Dacă z∈[-4, 4], f(z) reprezintă densitatea de probabilitate cu repartiţie normală normată din figura 19.3, pentru care s-a utilizat secvenţa MATLAB:

```
z=-4:.01:4;
f=1./(sqrt(2*pi))*exp(-z.^2/2);
plot(z,f); grid
```

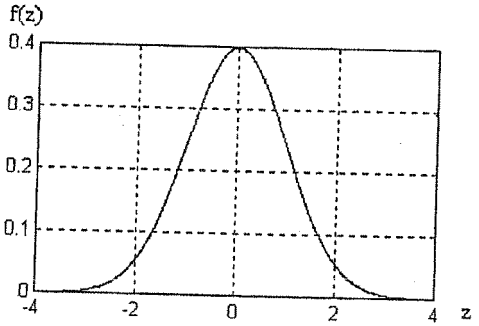


Fig.19.3. Densitatea de probabilitate cu repartiţie normală normată.

Calculul funcţiei de repartiţie Laplace:

$$F(z)=\int_{-\infty}^{z}f(z)dz=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z}e^{-\frac{t^2}{2}}dt$$

foloseşte secvenţa MATLAB:

```
F=trapz(z,1./(sqrt(2*pi))*exp(-z.^2/2))
```

Limita de integrare -∞ se va lua egală cu un număr între -5 şi -10, fără ca prin aceasta să se introducă erori semnificative.

Exemplu 19.5.1. Să se calculeze valoarea funcţiei de repartiţie normată (Laplace) pentru z = 1.5 şi pentru z∈[-1, 2.5].

Cu secvenţa MATLAB:

```
z1 = 1.5;  z = -10:.01:z1; z2 = [-1 2.5];
F1 = trapz(z,1./(sqrt(2*pi))*exp(-z.^2/2))
za = -10:.01:-1; zb=-10:.01:2.5;
Fa = trapz(za,1./(sqrt(2*pi))*exp(-za.^2/2));
Fb = trapz(zb,1./(sqrt(2*pi))*exp(-zb.^2/2));
F2 = Fb-Fa
```

se obţin rezultatele: F1 = 0.9332 şi F2 = 0.8351.

Partea a III-a
GRAFICĂ ÎN MATLAB

20. REPREZENTĂRI GRAFICE 2D

20.1. Reprezentări grafice elementare

Funcţiile MATLAB pentru reprezentări grafice elementare sunt:

plot Reprezintă grafice în coordonate X-Y liniare;
loglog Reprezintă grafice în coordonate X-Y logaritmice;
semilogx Reprezintă grafice în coordonate X-Y semilogaritmice (axa X este logaritmică);
semilogy Reprezintă graficele în coordonate X-Y semilogaritmice (axa Y este logaritmică);
fill Reprezintă grafic poligoane.

20.1.1. Reprezentarea grafică în coordonate liniare

Pentru reprezentarea datelor în coordonate liniare se utilizează funcţia *plot*; se apelează cu una dintre sintaxele:

plot(y) *plot*(x,y)
plot(x,y,'linie-tip') *plot*(x1,y1,x2,y2,...)

plot(y) - reprezintă grafic argumentul y funcţie de indici, cu următoarele precizări:
a) - dacă argumentul y este complex, *plot*(y) este echivalent cu *plot*(*real*(y),*imag*(y));
b) - dacă y este vector (linie sau coloană), funcţia *plot* trasează graficul y=y(i), unde i =1,2,...n este numărul de ordine al elementului y;
c) - dacă y este o matrice m x n, funcţia *plot* trasează graficele $y_j = y_j(i)$, unde i=1,2,...n este numărul de ordine al elementului de pe coloana j;

plot(x,y) - reprezintă grafic vectorul y funcţie de vectorul x, cu următoarele precizări:
a) - dacă x este vector, iar y este matrice, atunci coloanele lui y sunt trasate funcţie de vectorul x;
b) - dacă x şi y sunt matrice de aceeaşi dimensiune, se reprezintă coloanele lui y funcţie de coloanele lui x.

plot(x1,y1,x2,y2) - reprezintă simultan mai multe grafice în acelaşi sistem de coordonate.

Graficele se pot reprezenta utilizând linii, markere şi culori după codul din tabelul 20.1.

Pentru reprezentările grafice, se asociază fiecărei caracteristici un şir de 1-3 caractere, dintre cele menţionate în tabelul de mai sus. Aceste şiruri de caractere trebuie cuprinse între apostrofuri şi menţionate în combinaţia culoare-marker sau culoare-linie-tip. Dacă se precizează o singură caracteristică (marker, linie sau culoare), cea de-a doua este

selectată de calculator. Astfel, instrucţiunea *plot*(x1,y1,':',x2,y2,'+r') reprezintă cu „linie din puncte" pentru prima caracteristică (x1,y1), şi cu markere „plus" de culoare roşie, a doua caracteristică.

Tabelul 20.1.

LINII TIP		MARKERE TIP		CULORI	
continuă	-	plus	+	galben	y
întreruptă	--	steluţă	*	mov	m
puncte	:	cerc	o	albastru-deschis	c
linie-punct	-.	x	x	roşu	r
		punct	.	verde	i
				albastru	b
				alb	w
				negru	k

Dacă nu se specifică culoarea, MATLAB-ul foloseşte implicit galben. Pentru grafice multiple se utilizează succesiv primele şase culori din tabel.

Pentru o citire mai precisă a datelor pe grafic reprezentarea se poate asocia cu funcţia de trasare a caroiajului *grid*.

Funcţia *plot* returnează un vector coloană al identificatorilor de control al caracteristicilor obiectelor linie. Obiectele linie create cu *plot* sunt copii ai axelor curente.

Perechile (x,y) pot fi urmate de perechile parametrul/valoare, pentru a specifica proprietăţi suplimentare ale liniilor.

Exemplul 20.1.1. Să se reprezinte grafic funcţiile:

f(x)=sin(2π50t) cu linie-punct de culoare verde şi

g(x)=f(x)+.2 cu markere-stea de culoare roşie.

Cu secvenţa MATLAB:

```
t=0:.001:.02;
f=sin(2*pi*50*t);
g=f+.2;
plot(t,f,'-.g',t,g,'*r')
```

se obţine figura 20.1.

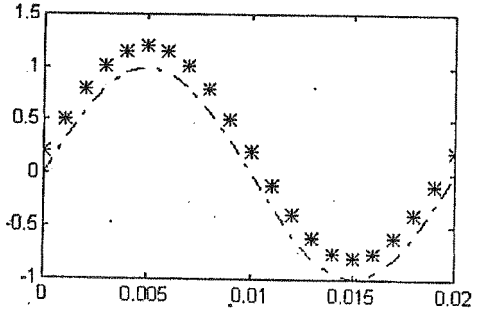


Fig. 20.1. Reprezentare grafică în coordonate liniare

20.1.2. Reprezentarea grafică în coordonate logaritmice şi semilogaritmice

Pentru reprezentările grafice în coordonate logaritmice sau semilogarimice se utilizează funcţiile *loglog*, *semilogx*, *semilogy;* se apelează cu sintaxele:

loglog(x,y) *semilogx*(x,y) *semilogy*(x,y)

Funcţia *loglog* scalează ambele axe utilizând logaritmul în baza 10, în timp ce funcţiile *semilogx* sau *semilogy* scalează logaritmic numai axa x, respectiv axa y, cealaltă axă fiind scalată liniar. Modul de utilizare al acestor funcţii este la fel ca acela al funcţiei *plot*.

Exemplul 20.1.2. Să se reprezinte în coordonate semilogaritmice (axa y) funcţia: $f(x)=10^x$.

Cu secvenţa MATLAB:

```
x=0:.1:10;  f=10.^x;  semilogy(x,f)
```

se obţine graficul din figura 20.2.

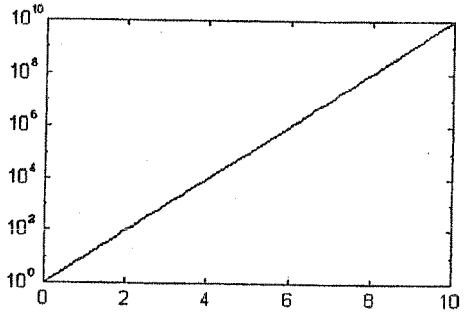


Fig. 20.2. Reprezentare grafică în coordonate semilogaritmice (axa y)

20.1.3. Reprezentarea grafică a poligoanelor

Reprezentarea grafică a poligoanelor utilizează funcţia *fill*; se apelează cu una dintre sintaxele:

fill(x,y,c) *fill*(x,y,'c') *fill*(x1,y1,c1,x2,y2,c2,...)

Funcţia *fill*(x,y,c) reprezintă un poligon definit de vectorii x şi y, cu nuanţele de culoare precizate de c. Coordonatele vârfurilor poligonului sunt specificate prin perechile (x,y).

Dacă argumentul c este un singur caracter dintre cele prezentate în lista de culori din tabelul 20.1 sau un vector cu trei componente [r g b], poligonul va fi colorat într-o singură culoare. În cazul în care c este un vector cu aceeaşi dimensiune ca x şi y, elementele acestuia sunt scalate cu funcţia *caxis* şi apoi utilizate ca indici într-o matrice care specifică culorile vârfurilor. Culorile dintre vârfuri sunt obţinute prin interpolare biliniară a culorilor vârfurilor.

Dacă x şi y sunt matrice de aceeaşi dimensiune, *fill*(x,y,c) reprezintă câte un poligon pentru fiecare coloană. În acest caz, c este un vector linie pentru poligoane cu o singură culoare şi respectiv o matrice pentru poligoane cu culori interpolate.

Dacă numai unul dintre argumentele x sau y este matrice, celălalt fiind vector coloană cu acelaşi număr de linii, vectorul coloană se va extinde la o matrice cu aceleaşi dimensiuni, prin adăugarea unor coloane identice.

Pentru specificarea poligoanelor multiple se poate utiliza şi forma *fill*(x1,y1,c1,x2,y2,c2,...), mult mai uşor de controlat.

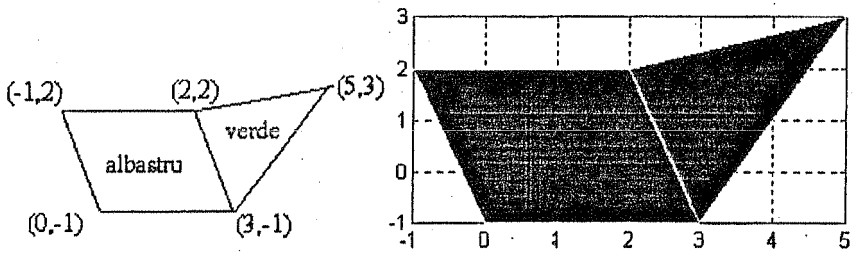
Funcţia *fill* setează proprietatea *FaceColor* a funcţiei *patch* (care creează module obiect) la *'flat'* (aceeaşi culoare), *'interp'* (culori interpolate) sau *ColorSpec* (culoarea specificată), funcţie de valorile matricei c.

Funcţia *fill* returnează un vector coloană al identificatorilor de control al caracteristicilor modulelor obiect, câte o linie pentru fiecare modul. Argumentele x,y,c pot fi urmate de perechi parametru/valoare pentru a specifica proprietăţi suplimentare ale modulului obiect.

Exemplul 20.1.3. Să se reprezinte grafic poligonul din figura 20.3a.
Cu secvenţa MATLAB:

```
x1=[0 3 2 -1];y1=[-1 -1 2 2];
x2=[3 5 2]; y2=[-1 3 2];
fill(x1,y1,'b',x2,y2, 'g')
```

se obţine rezultatul din figura 20.3b.



a. Modelul. b Reprezentarea grafică.
Fig. 20.3. Reprezentarea grafică a unui poligon.

20.2. Reprezentări grafice speciale X-Y

Funcţiile MATLAB pentru reprezentări grafice speciale sunt:

polar Reprezintă grafice în coordonate polare;
bar Reprezintă grafice cu bare;
stem Reprezintă grafice sub formă discretă (util pentru semnale discrete);
stairs Reprezintă grafice în trepte (util pentru semnale cuantizate);
errorbar Evidenţiază eroarea datelor reprezentate grafic;
hist Reprezintă grafic histograma datelor;
rose Reprezintă grafic histograma unghiulară a datelor (coordonate polare);
compass Reprezintă grafic vectorii argument, precizaţi prin proiecţiile pe axe, cu săgeţile orientate dinspre origine;
feather Reprezintă grafic vectorii argument, precizaţi prin proiecţiile pe axe, ordonaţi echidistant pe axa orizontală;

fplot Realizează controlul reprezentărilor grafice cu parametri impuşi;
comet Reprezintă dinamic (în mişcare) traiectoria unui punct într-o reprezentare grafică 2D.

20.2.1. Reprezentarea grafică în coordonate polare

Reprezentarea în coordonate polare se face cu funcţia *polar*, se apelează cu una dintre sintaxele:

polar(theta,r) *polar*(theta, r, 'linie-tip')

Modul de folosire al opţiunii linie-tip este identic cu cel al funcţiei *plot*.

Exemplul 20.2.1. Să se reprezinte în coordonate polare funcţia:

f(x)=sin(2t)cos(2t).

Cu secvenţa MATLAB:

```
t=0:.01:2*pi;
f=sin(2*t).*cos(2*t);
polar(t,f)
```

se obţine reprezentarea din figura 20.4.

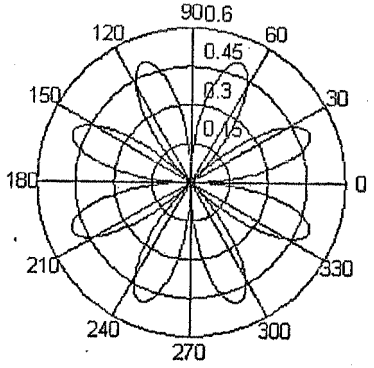


Fig. 20.4. Reprezentare în coordonate polare.

20.2.2. Reprezentarea graficelor cu bare

Reprezentarea grafică cu bare se face cu funcţia *bar*, se apelează cu una dintre sintaxele:

bar(y) - trasează un grafic de bare cu elementele vectorului y, adică $y=y_i$.

bar(x,y) - trasează un grafic de bare cu elementele vectorului y la locaţiile specificate de vectorul x, adică y=y(x). Valorile lui x trebuie să fie egal depărtate şi crescătoare.

[xb,yb] = *bar*(y), şi
[xb,yb] = *bar*(x,y) - nu reprezintă graficele, dar calculează vectorii xb şi yb astfel încât *plot*(xb,yb) să poată trasa graficul de bare. Aceasta e utilă în situaţiile în care se doreşte un control mai mare asupra graficului;

spre exemplu, combinarea mai multor grafice de bare elaborate cu instrucţiunea *plot*.

Exemplul 20.2.2. Să se reprezinte graficul cu bare al datelor conţinute în vectorul y=[1 3 7 6 5 2 3].
Cu secvenţa MATLAB:

```
y=[1 3 7 6 5 2 3]; bar(y)
```

se obţine graficul din figura 20.5.

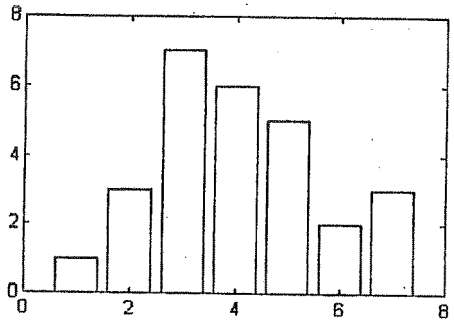


Fig. 20.5. Reprezentare grafică cu funcţia *bar*.

20.2.3. Reprezentarea discretă a datelor

Reprezentarea grafică a semnalelor discrete se face cu funcţia *stem*, sub forma unor linii terminate cu cerculeţ la extremitatea opusă axei; se apelează cu una dintre sintaxele:

stem(y) - trasează un grafic din linii cu cerculeţ, cu elementele vectorului y;

stem(x,y) - trasează un grafic din linii terminate cu cerculeţ, cu locaţiile specificate de vectorul x, adică y=y(x). Valorile lui x trebuie să fie egal depărtate şi crescătoare.

stem(x,y, linie_tip) - trasează un grafic din linii de tipul şi culoarea precizată în şirul de caractere linie_tip, aşa cum a fost precizat la funcţia plot. Spre exemplu: stem(x,y,':r').

Exemplul 20.2.3. Să se reprezinte grafic funcţia discretă sinus:

$$f[n] = \sin\left(\frac{2\pi}{10}n\right), \qquad n \in [0,20]$$

Cu secvenţa MATLAB:

```
n=0:20; f=sin(2*pi*n/10); stem(n,f)
```

se obţine graficul din figura 20.6.

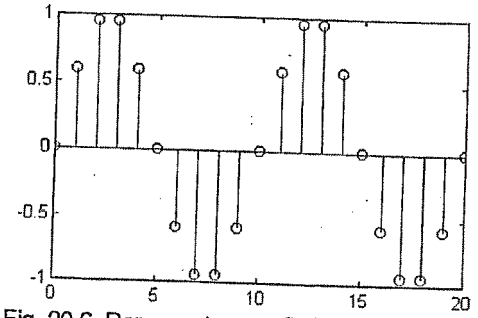


Fig. 20.6. Reprezentare grafică cu funcţia *stem*.

20.2.4. Reprezentarea graficelor în trepte

Graficele în trepte sunt utilizate la reprezentarea diagramelor sistemelor numerice de eşantionare şi prelucrare a datelor.

Reprezentarea grafică în trepte se face cu funcţia *stairs*, care se apelează cu una dintre sintaxele:

stairs(y) - trasează graficul în trepte al elementelor vectorului y.

stairs(x,y) - trasează graficul în trepte al elementelor vectorului y la locaţiile specificate în x. Valorile lui x trebuie să fie egal depărtate şi în ordine crescătoare.

[xb,yb] = *stairs*(y), şi
[xb,yb] = *stairs*(x,y) - calculează vectorii xb şi yb, astfel încât *plot*(xb,yb) să poată trasa graficul în trepte.

Exemplul 20.2.4. Să se reprezinte graficul în trepte al funcţiei y=sin(x).
Cu secvenţa MATLAB:

```
x = 0:.2:6; y=sin(x); stairs(x,y)
```

se obţine graficul din figura 20.7.

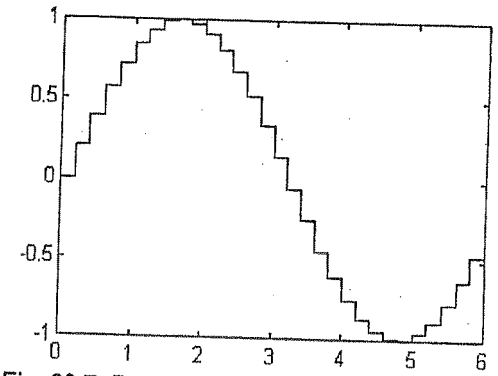


Fig. 20.7. Reprezentare grafică cu funcţia *stairs*.

20.2.5. Reprezentarea grafică a erorilor

Reprezentarea grafică a datelor cu bare de eroare ataşează fiecărei perechi (x,y) eroarea precizată într-un vector cu aceleaşi dimensiuni; se apelează cu sintaxa:

errorbar(x,y,e)

Vectorul e conţine lungimea barelor ce reprezintă eroarea. „Barele de erori" se reprezintă simetric în raport cu ordonata y, ceea ce presupune o asociere de erori pozitive sau negative, cu aceeaşi probabilitate. Segmentele de eroare sunt de înălţime 2*e şi se trasează pe curba y=y(x). Figura obţinută reprezintă plaja de valori pe care o poate lua funcţia y cu eroarea „e".

Dacă x şi y sunt matrice de aceeaşi dimensiune, funcţia *errorbar* va reprezenta graficul cu bare de eroare pentru fiecare coloană în parte.

Exemplul 20.2.5. Să se reprezinte un grafic cu bare de erori pentru:

y=sin(x).

Cu secvenţa MATLAB:

```
x = 0:.2:6; y = sin(x);
e=rand(size(x))/5;
errorbar(x,y,e)
```

se obţine reprezentarea din figura 20.8.

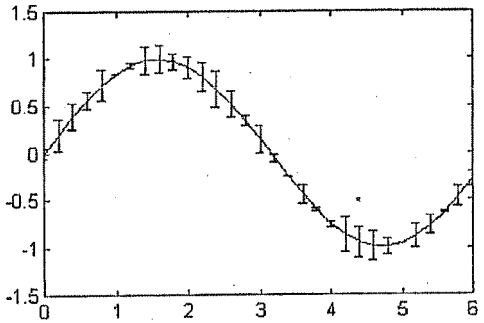


Fig. 20.8. Reprezentare grafică cu funcţia *errorbar*.

20.2.6. Reprezentarea grafică a histogramelor

Calculul şi reprezentarea grafică a histogramelor se face cu funcţia *hist*; se apelează cu una dintre sintaxele:

hist(y) - trasează histograma cu 10 segmente a datelor vectorului y;
hist(y,nb) - trasează histograma cu nb segmente a datelor vectorului y;
hist(y,x) - trasează histograma datelor vectorului y la abscisele specificate în x;
[n,x]=*hist*(y),
[n,x]=*hist*(y,nb),
[n,x]=*hist*(y,x) - returnează vectorii n şi x conţinând frecvenţa de apariţie şi localizarea segmentelor; cu *bar*(x,n) se poate apoi trasa histograma. Aceste proceduri se utilizează în situaţiile în care este necesară o mai mare

flexibilitate în reprezentările grafice. Combinarea histogramei cu instrucţiuni mai elaborate de trasare şi reprezentare grafică poate conduce la obţinerea unor grafice deosebit de sugestive.

Exemplul 20.2.6. Să se genereze histograma unui vector cu elementele distribuite normal (Gaussian).

Cu secvenţa MATLAB:

```
x=-3:.3:3; y=randn(10000,1); hist(y,x)
```

se obţine graficul din figura 20.9.

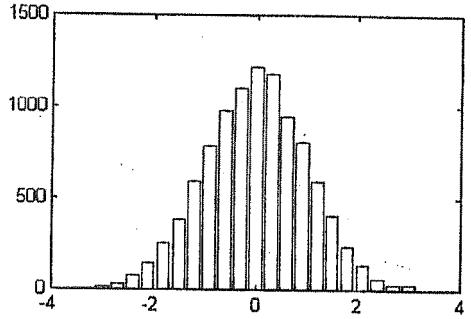


Fig. 20.9. Reprezentare grafică a unei histograme.

Reprezentarea unei histograme în coordonate polare se face cu funcţia *rose*; se apelează cu una dintre sintaxele:

rose(x) *rose*(x,N) [t,r]=*rose*(...)

unde x trebuie să fie în intervalul [0, 2π], iar N este numărul de subintervale în care se împarte intervalul [0, 2π]. Valoarea implicită pentru N este 20.

Dacă funcţia *rose* se apelează cu argumente, atunci returnează cei doi vectori t şi r, care se vor folosi pentru reprezentarea histogramei *polare*, cu funcţia *polar*(t,r).

Exemplul 20.2.7. Să se reprezinte grafic histograma a 100 numere aleatoare în coordonate polare.

Cu secvenţa MATLAB:

```
x=2*pi*rand(100,1); rose(x,10)
```

se obţine graficul din figura 20.10.

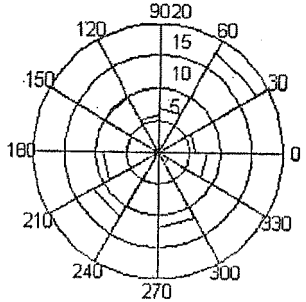


Fig. 20.10. Reprezentare grafică a histogramei cu funcţia *rose*.

20.2.7. Reprezentarea grafică a vectorilor

Funcţia *compass* reprezintă grafic vectori cu originea în originea sistemului de coordonate; se apelează cu una dintre sintaxele:

compass(z) *compass*(x,y)

unde z este numărul complex x+iy, iar x şi y sunt numere reale - proiecţia vectorului pe abscisă şi ordonată.

Exemplul 20.2.8. Să se reprezinte grafic vectorii:

$z1 = 2 - 5i$ $z2 = -2 + i$ $z3 = 3 + 2i$

Cu secvenţa MATLAB:

```
z=[2-5*i, -2+i, 3+2*i];
compass(z); grid
```

se obţine graficul din figura 20.11:

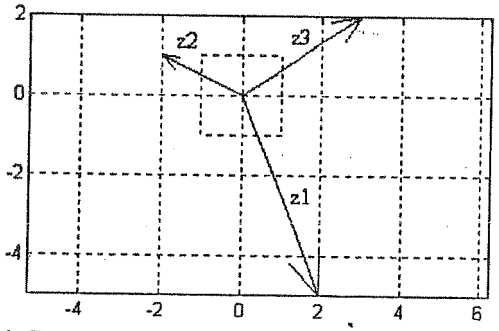


Fig. 20.11. Reprezentare grafică a vectorilor cu funcţia *compass*.

Funcţia *feather* reprezintă grafic vectori cu originea plasată echidistant pe axa Ox; se apelează cu una dintre sintaxele:

feather(z) *feather*(x,y)

unde z este numărul complex x+iy, iar x şi y sunt numere reale - proiecţia vectorului pe abscisă şi ordonată.

Exemplul 20.2.9. Să se reprezinte grafic cu originile pe axa Ox vectorii:

$z1 = 2 - 5i$ $z2 = -2 + i$ $z3 = 3 + 2i$

Cu secvenţa MATLAB:

```
z=[2-5*i, -2+i, 3+2*i];
feather(z); grid
```

se obţine graficul din figura 20.12.

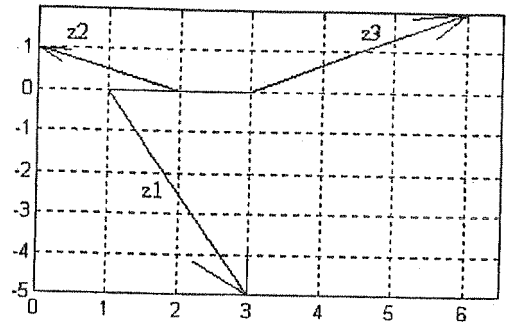


Fig. 20.12. Reprezentare grafică a vectorilor cu funcţia *feather*.

20.2.8. Reprezentări grafice cu parametri impuşi

Funcţia *fplot* realizează o reprezentare grafică cu anumite restricţii; se apelează cu una dintre sintaxele:

fplot('fun', limite) *fplot*('fun', limite,n,unghi)
fplot('fun', limite,n) *fplot*('fun', limite,n,unghi,subdiv)
[x, y]= *fplot*('fun', limite,...)

Funcţia *fplot* reprezintă grafic funcţia „fun" între limitele specificate de argumentul „limite".

Argumentele funcţiei *fplot* au următoarele semnificaţii:

fun - numele fişierului funcţie (şir de caractere);
limite=[xmin xmax] - limitele axei x pentru care se doreşte reprezentarea grafică;
n - numărul de eşantioane cu care este reprezentată funcţia (implicit n=25);
unghi - cea mai mare schimbare de unghi dintre două segmente adiacente ale graficului (implicit 10°);
subdiv - numărul maxim de subdiviziuni dintre două diviziuni ale scalei pentru care unghiul dintre două segmente adiacente nu este mai mare decât valoarea impusă (implicit 20).

Dacă funcţia *fplot* se apelează cu argumentele de ieşire [x,y], în aceşti vectori coloană sunt returnate valorile abscisei şi ordonatei funcţiei. Reprezentarea grafică se poate face ulterior cu funcţia *plot*(x,y).

Exemplul 20.2.9. Fie fişierul funcţie:

```
function y=test(x)
y=sin(x)./x;
```

înregistrat cu numele **test.m**. Să se reprezinte grafic funcţia din fişierul **test.m**, între limitele [-20, 20] cu n=50 eşantioane.

Cu secvenţa MATLAB:

```
fplot('test',[-20 20], 50); grid
```

se obţine graficul din figura 20.13.

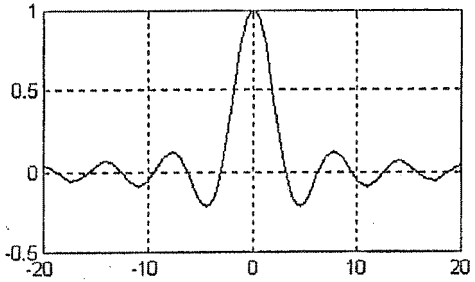


Fig. 20.13. Reprezentare grafică realizată cu funcţia *fplot*.

20.2.9. Reprezentări grafice dinamice 2D

Pentru a reprezenta grafic un punct care urmăreşte realizarea grafică („cometă") se foloseşte funcţia *comet*; se apelează cu una dintre sintaxele:

comet(y) *comet*(x,y) *comet*(x,y,p)

Apelată fără argumente de intrare, funcţia *comet* lansează demonstraţia cu acelaşi nume. Dacă se apelează cu unul sau cu două argumente de intrare, se realizează o reprezentare grafică a lui y sau y(x), urmărite de un marker („cometă"), la o distanţă de 0.1 din lungimea vectorului y. Apelarea cu trei argumente realizează acelaşi lucru, însă distanţa de urmărire este setată la valoarea p din lungimea totală a vectorului y.

Exemplul 20.2.9. Urmăriţi modul de lucru al funcţiei comet, cu secvenţa MATLAB:

```
t = 0:1/10000:.1; y=sin(2*pi*50*t);
comet(t,y,.2)
```


21. REPREZENTĂRI GRAFICE 3D

În acest capitol se prezintă funcţiile MATLAB pentru reprezentări grafice în plan sau în spaţiu asociate câmpurilor bidimensionale:

clabel	Plasează marcaje pe liniile de contur, referitoare la cota Z;
comet3	Reprezintă dinamic (în mişcare) traiectoria unui punct, într-o reprezentare grafică 3D;
contour	Reprezintă grafic în 2D liniile de contur (liniile de nivel constant);
contourc	Returnează o matrice care conţine perechile (cota Z - număr puncte şi coordonata X - coordonata Y);
contour3	Reprezintă grafic în 3D liniile de contur (liniile de nivel constant);
fill3	Reprezintă grafic poliedre în 3D;
plot3	Reprezintă grafice în 3D;
quiver	Reprezintă grafic orientarea unui câmp de vectori.

21.1. Reprezentarea liniilor de contur

Prin linii de contur se înţeleg liniile situate la acelaşi nivel pe axa Z (linii de nivel constant).

21.1.1. Calculul matricei liniilor de contur

Matricea care conţine perechile de coordonate (X,Y) ale fiecărei linii de contur, se determină cu funcţia *contourc*; se apelează cu una dintre sintaxele:

C=*contourc*(Z)	- calculează matricea liniilor de contur ale matricei Z. Numărul liniilor de nivel şi valorile acestora sunt alese automat;
C= *contourc*(Z,n)	- calculează matricea liniilor de contur ale matricei Z pentru n linii de contur (n este un scalar);
C= *contourc*(Z,v)	- calculează matricea liniilor de contur ale matricei Z la nivelele specificate de vectorul v;
C= *contourc*(x,y,Z), C= *contourc*(x,y,Z,n) şi C= *contourc*(x,y,Z,v)	- calculează matricea liniilor de contur ale matricei Z şi utilizează date din vectorii x şi y pentru a controla scalarea axelor Ox şi Oy. Elementele x şi y sunt cu pas constant.

Matricea liniilor de contur, C, este o matrice cu două linii, ca în exemplul:

C=[nivel1 x1 x2 x3 ... nivel2 x1 x2 x3 ...
perechi1 y1 y2 y3 ... perechi2 y1 y2 y3 ...]

Vectorii x şi y trebuie să fie monoton crescători şi cu pas constant.

21.1.2. Reprezentarea grafică în plan a liniilor de contur

Reprezentarea grafică în plan a liniilor de contur face apel la funcţia *contour*; se apelează cu una dintre sintaxele:

contour(Z) - reprezintă conturul liniilor de acelaşi nivel ale matricei Z. Colţul din stânga sus al desenului corespunde elementului Z(1,1). Numărul liniilor conturului şi valorile cotelor Z sunt alese automat;

contour(Z,N) - reprezintă conturul a n linii de nivel ale matricei Z. Dacă N nu este specificat, valoarea implicită este 10;

contour(Z,V) - reprezintă conturul liniilor de nivel ale matricei Z, caracterizate de valorile specificate în vectorul V;

contour(X,Y,Z,N), şi
contour(X,Y,Z,V) - reprezintă conturul liniilor de nivel ale matricei Z, utilizând;
N - pentru numărul liniilor de nivel,
V - pentru valorile liniilor de nivel,
X - pentru controlul scalării pe axa Ox,
Y - pentru controlul scalării pe axa Oy.
Dacă scalele nu se precizează (vectorii x şi y lipsesc), reprezentarea se face după numărul de elemente în care au fost divizate axele x şi y.

contour (...,'tip-linie') - reprezintă liniile de contur cu liniile şi culorile specificate;

C=*contour*(...) - returnează matricea liniilor de contour, C, descrisă la funcţia *contourc* şi este utilizată de funcţia *clabel*;

[C,H]=*contour*(...) - returnează matricea liniilor de contour, C, şi un vector coloană H al identificatorilor obiectelor linie.

Matricea Z trebuie să aibă cel puţin dimensiunea 3.

Exemplul 21.1.1. Să se reprezinte grafic conturul liniilor de nivel ale funcţiei $z=x\exp(-x^2-y^2)$ în domeniul: $-2 \le x \le 2$, $-2 \le y \le 3$

Cu secvenţa MATLAB:

```
[x,y] = meshdom(-2:.2:2, -2:.2:3);
z = x.*exp(-x.^2-y.^2);
contour(-2:.2:2,-2:.2:3,z,8)
```

se obţine reprezentarea din figura 21.1.

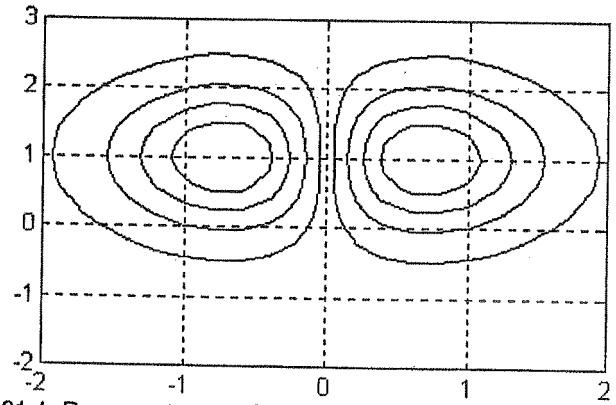


Fig. 21.1. Reprezentare grafică a liniilor de contur cu funcţia *contour*.

21.1.3. Etichetarea cotelor liniilor de contur

Pentru a preciza cotele (nivelul) liniilor de contur într-o reprezentare grafică se foloseşte funcţia *clabel*; se apelează cu una dintre sintaxele:

clabel(C) - etichetează liniile de nivel. Poziţia acestora este aleasă aleator;

clabel(C,V) - etichetează liniile de nivel precizate de vectorul V;

clabel(C,'manual') - etichetează liniile de nivel selectate cu mouse-ul. Se apasă tasta „ENTER" pentru a termina acţiunea şi „Space Bar" pentru a introduce următoarea linie de nivel.

Exemplul 21.1.3. Să se reprezinte grafic şi să se eticheteze liniile de nivel ale funcţiei $z=x\exp(-x^2-y^2)$ în domeniul: $-2 \le x \le 2$, $-2 \le y \le 3$

Cu secvenţa MATLAB:

```
[x,y] = meshdom(-2:.2:2, -2:.2:3);
z = x.*exp(-x.^2-y.^2);
C=contour(-2:.2:2,-2:.2:3,z,8)
clabel(C);
```

se obţine graficul din figura 21.2.

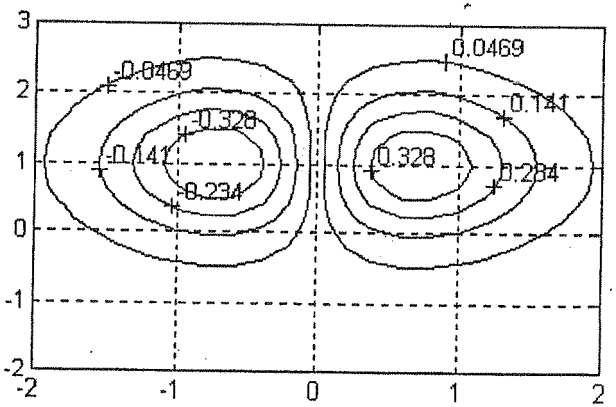


Fig. 21.2. Etichetarea liniilor de nivel cu funcţia *clabel*.

21.1.4. Reprezentarea grafică în spaţiu a liniilor de contur

Reprezentarea grafică în spaţiu a liniilor de nivel constant se realizează cu funcţia *contour3*; se apelează cu una dintre sintaxele:

contour3(Z) - realizează reprezentarea 3D a liniilor de contur ale matricei Z;

contour3(Z,N) - realizează reprezentarea 3D a N linii de contur ale matricei Z;

contour3(X,Y,Z) şi
contour3(X,Y,Z,N) - utilizează matricele X şi Y pentru a defini limitele axelor.

Vectorii X şi Y trebuie să fie monoton crescători şi cu pas constant.

Exemplul 21.1.4. Să se reprezinte grafic în 3D liniile de contur ale funcţiei predefinite *peaks*.
Cu secvenţa MATLAB:

```
x=-3:.125:3;
y=x;
[X,Y]=meshgrid(x,y);
Z=peaks(X,Y);
contour3(X,Y,Z,20)
```

se obţine secvenţa din figura 21.3.

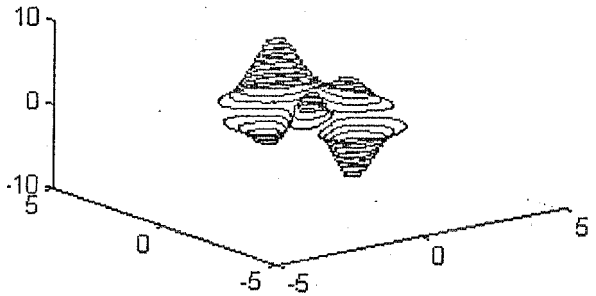


Fig. 21.3. Reprezentare grafică a liniilor de contur cu funcţia *contour3*.

21.1.5. Reprezentarea grafică a câmpurilor de vectori orientaţi

Reprezentarea grafică a unui câmp de vectori orientaţi foloseşte funcţia *quiver*, se apelează cu una dintre sintaxele :

quiver(X,Y,DX,DY) - reprezintă mici segmente de dreaptă cu săgeţi (vectori) având originea la perechile de elemente (X,Y). Fiecare pereche de elemente din matricele DX şi DY sunt proiecţiile vectorului pe axele Ox şi Oy;

quiver(DX,DY) - presupune implicit X=1:n şi Y=1:m. În acest caz DX şi DY sunt pe o reţea rectangulară;

quiver(x,y,dx,dy,s) şi
quiver(dx,dy,s) - controlează lungimea săgeţilor prin factorul de scală s.

Culorile şi tipurile de linii care se pot folosi sunt cele precizate la funcţia *plot*.

Exemplul 21.1.5. Să se reprezinte grafic vectorii unei mişcări browniene.
Cu secvenţa MATLAB:

```
xg = -2:.5:2;
yg = -2:.5:2;
dx=rand(length(xg),length(xg))-0.5;
dy=rand(length(yg),length(yg))-0.5;
quiver(xg,yg,dx,dy)
```

se obţine reprezentarea din figura 21.4.

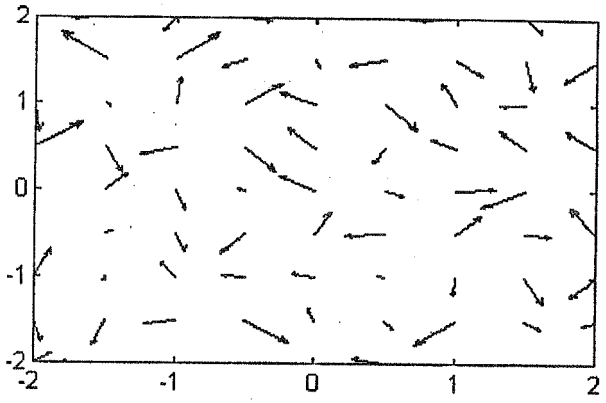


Fig. 21.4. Reprezentare grafică a câmpurilor de vectori cu funcţia *quiver*.

21.2. Reprezentări spaţiale cu linii

21.2.1. Reprezentarea liniilor în spaţiu

Reprezentarea liniilor în spaţiu se face cu funcţia *plot3*, care se apelează cu una dintre sintaxele:

plot3(x,y,z) - unde x, y şi z sunt vectori de aceeaşi dimensiune, reprezintă grafic o linie în spaţiul 3D, linie care trece prin punctele ale căror coordonate sunt tripletele (x,y,z);

plot3(X,Y,Z) - unde X, Y şi Z sunt matrice de aceleaşi dimensiuni, reprezintă grafic câte o linie în spaţiul 3D, pentru fiecare triplet al coloanelor matricelor [X(:,i),Y(:,i),Z(:,i)];

plot3(x,y,z,'linie-tip'), sau
plot3(x1,y1,z1,'linie-tip1',x2,y2,z2,'linie-tip2',...) - realizează reprezentări grafice 3D multiple, în care se precizează tipurile de linii şi culorile acestora, ca în cazul funcţiei *plot*.

Funcţia *plot3* returnează un vector coloană al identificatorilor de control al caracteristicilor obiectelor linie. Obiectele linie create cu *plot3* sunt copii ai axelor curente. Tripletele (x,y,z) pot fi urmate de perechile parametrul/valoare, pentru a specifica proprietăţi suplimentare ale liniilor.

Pentru o citire mai precisă pe grafic, reprezentarea se poate asocia cu funcţia *grid*.

Exemplul 21.2.1. Să se reprezinte grafic o spirală în 3D.
Cu secvenţa MATLAB:

```
t=0:pi/50:10*pi;
plot3(sin(t), cos(t),t)
```

se obţine figura 21.5.

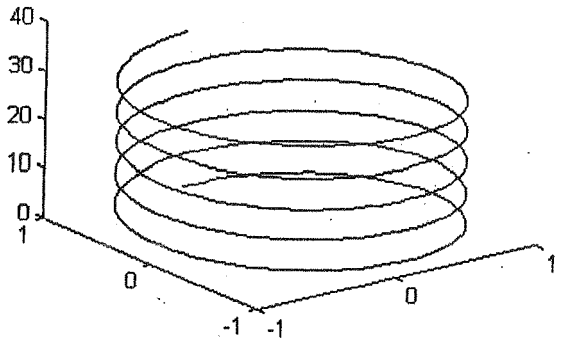


Fig. 21.5. Reprezentarea grafică a unei linii în spaţiu cu funcţia *plot3*.

21.2.2. Reprezentarea grafică spaţială a poliedrelor

Reprezentarea grafică în spaţiu a poliedrelor se face cu funcţia *fill3*, care se apelează cu una dintre sintaxele:

fill3(x,y,z,c) *fill3*(x,y,z,'c') *fill3*(x1,y1,z1,c1,x2,y2,z2,c2,...)

Funcţia *fill3*(x,y,z,c) reprezintă un poliedru 3D, având vârfurile definite de vectorii x, y şi z, cu nuanţele de culoare precizate de c. Coordonatele poliedrului sunt specificate prin tripletele (x,y,z).

Dacă c este un singur caracter dintre cele prezentate în lista de culori de la funcţia plot: r, g, b, c, m, y, w, m, k sau un vector cu trei componente [r g b], poliedrul va fi colorat cu o singură culoare. Dacă c este un vector cu aceeaşi lungime ca x şi y, elementele acestuia sunt scalate cu funcţia *caxis* şi apoi utilizate ca indici în matricea de culoare pentru a specifica culorile vârfurilor. Culorile dintre vârfuri sunt obţinute prin interpolare biliniară a culorilor vârfurilor.

Dacă x, y şi z sunt matrice de aceeaşi dimensiune, *fill3*(x,y,z,c) reprezintă câte un poliedru pentru fiecare coloană. În acest caz, c este un vector linie pentru poliedre reprezentate cu o singură culoare şi este o matrice pentru poliedre realizate cu culori interpolate (efecte speciale).

Dacă numai unul din argumentele x, y sau z sunt matrice, celelalte fiind vectori coloană cu acelaşi număr de linii, vectorii coloană se extind la matrice cu aceleaşi dimensiuni, prin adăugarea unor coloane identice.

Pentru specificarea poliedrelor multiple se poate utiliza şi forma *fill3*(x1,y1,z1,c1,x2,y2,z2,c2,...), mult mai uşor de controlat.

Funcţia *fill3* returnează un vector coloană al identificatorilor de control al modulelor obiect, câte o linie pentru fiecare modul. Argumentele x,y,z,c pot fi urmate de perechile parametru/valoare pentru a specifica proprietăţi suplimentare ale modulului obiect.

21.2.3. Reprezentări grafice spaţiale dinamice

Reprezentarea grafică a unui punct care urmăreşte realizarea grafică spaţială („cometă") se face cu funcţia *comet3*; se apelează cu una dintre sintaxele:

comet3(z) *comet3*(x,y,z) *comet3*(x,y,z,p)

Apelată fără argumente, funcţia *comet3* lansează demonstraţia cu acelaşi nume. Dacă se apelează cu unul sau cu trei argumente, se obţine o reprezentare grafică a lui z sau a tripletelor (x,y,z), urmărite de un marker („cometă"), la o distanţă de 0.1 din lungimea vectorului z. Apelarea cu patru argumente realizează acelaşi lucru, însă distanţa de urmărire este setată la valoarea p din lungimea totală a vectorului z.

Exemplul 21.2.2. Urmăriţi modul de lucru al funcţiei *comet3*, cu secvenţa MATLAB:

```
t = 0:1/100:20;comet3(sin(t),cos(t),t)
```

21.3. Reprezentarea 3D a suprafeţelor şi liniilor de contur

Funcţiile MATLAB folosite pentru reprezentări 3D ale suprafeţelor şi liniilor de contur sunt:

mesh	Reprezintă grafic suprafeţe 3D sub forma unei „reţele" („mesh");
meshc	Reprezintă grafic combinaţia suprafaţă 3D („mesh")/linii de contur (reprezentate sub suprafaţă);
meshz	Reprezintă grafic suprafeţe 3D („mesh"), cu plan de referinţă la cota zero (piedestal);
surf	Reprezintă grafic suprafeţe pline 3D;
surfc	Reprezintă grafic combinaţia suprafaţă 3D (continuă)/linii de contur (reprezentate sub suprafaţă).

O suprafaţă este parametrizată prin două variabile independente, i şi j, care variază continuu în interiorul unui dreptunghi; spre exemplu $1 \le i \le m$ şi $1 \le j \le n$. În aceste condiţii, fiecare punct este specificat prin trei funcţii, X(i,j), Y(i,j) şi Z(i,j). Dacă i şi j sunt numere întregi, atunci suprafaţa este definită în nodurile unei reţele rectangulare (grid), matricele având dimensiunea m×n. Dacă se doreşte şi precizarea culorii suprafeţei, atunci mai este necesară încă o matrice, C(i,j), de aceeaşi dimensiune, m×n. Fiecare punct al suprafeţei rectangulare este conectat cu alte patru puncte vecine.

$$\begin{array}{ccccc} & & i-1,j & & \\ & & | & & \\ i,j-1 & - & i,j & - & i,j+1 \\ & & | & & \\ & & i+1,j & & \end{array}$$

Punctele de pe margini sunt conectate cu trei noduri, iar colţurile au numai două noduri vecine.

Culoarea suprafeţei poate fi specificată prin două metode diferite: specificând colţurile reţelei sau centrele acesteia. Dacă funcţia *shading*, care configurează nuanţele, este iniţializată la „interp", C trebuie să aibă aceeaşi dimensiune ca matricele X,Y şi Z; ea specifică culorile colţurilor, în interiorul zonei realizându-se o interpolare liniară. Dacă *shading* este iniţializată „faceted" (implicit) sau „flat", atunci C(i,j) specifică o culoare constantă în dreptunghiul respectiv :

$$\begin{array}{ccc} (i,j) & - & (i,j+1) \\ | & C(i,j) & | \\ (i+1,j) & - & (i+1,j+1) \end{array}$$

În acest caz C poate avea aceeaşi dimensiune cu X,Y şi Z, însă ultima linie şi coloană sunt ignorate, sau poate fi dimensionată cu o linie şi o coloană mai puţin decât acestea.

Reprezentarea grafică 3D a suprafeţelor se poate face fie sub forma unei „reţele" („mesh") , fie sub forma suprafeţelor netede.

21.3.1. Reprezentarea suprafeţelor cu „mesh"

Reprezentarea suprafeţelor cu „mesh" se face folosind funcţiile: *mesh*, *meshc* şi *meshz* care se apelează cu sintaxele:

mesh(X, Y, Z, C) *mesh*(X, Y, Z)
mesh(Z) *mesh*(Z,C)
meshc(...) *meshz*(...)

În cazul cel mai general funcţia *mesh* se apelează cu patru matrice ca parametri de intrare; reprezintă grafic suprafaţa Z(X,Y), cu culorile din matricea C.

În cele mai multe aplicaţii X şi Y sunt vectori. Aceştia trebuie să fie ordonaţi crescător şi cu pas constant, pentru a rezulta o figură corectă. Dacă argumentele X şi Y sunt omise, reprezentarea este făcută pe baza indicilor matricei Z.

În cazul în care matricea C este omisă, se consideră C=Z, astfel încât culoarea este proporţională cu înălţimea suprafeţei.

Poziţia din care este observată suprafaţa reprezentată grafic poate fi precizată cu funcţia *view*. Gradarea axelor este dată de intervalele X, Y şi Z, sau de setarea curentă a axelor, prin funcţiile *axis* sau *axes*. Culorile utilizate sunt cele determinate de C sau precizate prin funcţia *caxis*. Valorile scalei de culori sunt utilizate ca indici ai unui tabel de culori.

Funcţia *meshc* permite reprezentarea 3D a suprafeţelor, cu „mesh", la care se asociază liniile de contur, trasate ca proiecţii în planul bazei; se apelează cu aceleaşi argumente ca funcţia *mesh*.

Funcţia *meshz* permite reprezentarea 3D a suprafeţelor, cu „mesh", trasând în plus un plan de referinţă la valoarea minimă (piedestal); se apelează cu aceleaşi argumente ca funcţia *mesh*.

Funcţia *mesh* returnează un identificator spre obiectul suprafaţă.

Funcţiile *axis*, *caxis*, *colormap*, *hold*, *shading* şi *view* setează proprietăţile obiectelor figură, axe şi suprafaţă, care afectează suprafeţele „mesh" afişate.

Exemplul 21.3.1. Reprezentaţi grafic funcţia $Z=X\exp(-X^2-Y^2)$:

1. ca suprafaţă „mesh",
2. ca suprafaţă „mesh" asociată cu linii de contur şi
3. ca suprafaţă „mesh" cu plan de referinţă.

Cu secvenţa MATLAB:

```
[X,Y]=meshgrid (-2:.2:2, -2:.2:2);
Z=X.*exp(-X.^2-Y.^2);
subplot(221);  mesh(X,Y,Z)
subplot(222); meshc(X,Y,Z)
subplot(223); meshz(X,Y,Z)
```


se obţin reprezentările din figura 21.6.

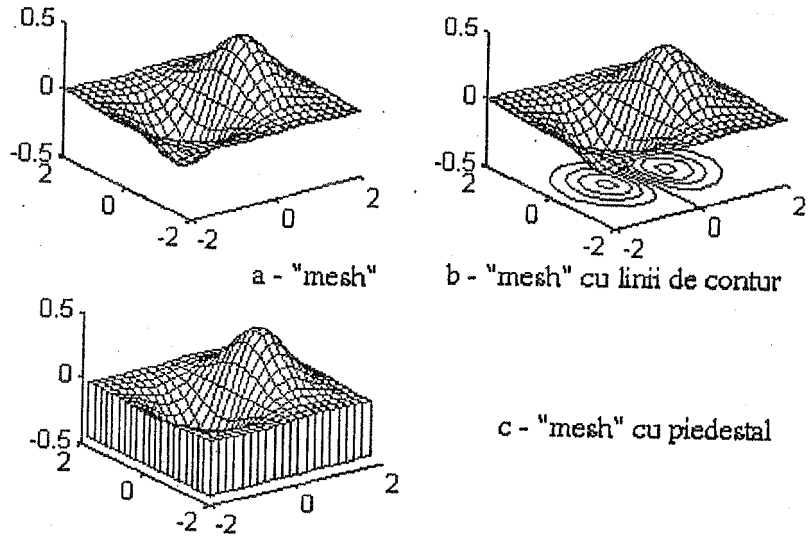


Figura 21.6. Reprezentare cu „mesh" a suprafeţelor 3D.

21.3.2. Reprezentarea grafică a suprafeţelor netede

Funcţiile *surf* şi *surfc* reprezintă suprafeţe 3D, sau suprafeţe 3D asociate cu liniile de nivel proiectate pe planul bazei; se apelează cu una dintre sintaxele :

surf(X,Y,Z,C) - reprezintă o suprafaţă descrisă de matricele X,Y şi Z, şi colorată cu elementele precizate în matricea C. În utilizări simple argumentele X şi Y pot fi vectori sau pot fi omise; argumentul C poate fi şi el omis;

Punctul din care este văzută o reprezentare 3D poate fi precizat prin funcţia *view*. Axele sunt date de matricile X,Y şi Z sau setate cu funcţia *axis*. Culoarea este dată de matricea C sau este setată prin funcţia *caxis*. Valorile scalate ale culorilor sunt indici la matricea de culoare curentă.

surf(X,Y,Z) - consideră C=Z, astfel încât culoarea este proporţională cu înălţimea suprafeţei;

surf(x,y,Z) şi
surf(x,y,Z,C)- realizează reprezentarea suprafeţei descrise de matricea Z (mxn), unde vectorul x are dimensiunea n, vectorul y are dimensiunea m, iar matricea C are dimensiunea m × n. În acest caz colţurile dreptunghiurilor care compun suprafaţa sunt tipletele (x(j), y(i), Z(i,j)). Prin urmare x este asociat numărului de coloane, iar y numărului de linii;

surf(Z) şi
surf(Z,C) - presupune x=1:n şi y=1:m pentru reprezentarea grafică 3D a matricei Z, folosind eventual şi matricea de culoare C;

surfc(...) - este identică cu *surf*(...), exceptând liniile de nivel, care sunt reprezentate sub suprafaţă, pe acelaşi grafic.

Funcţiile *surf* şi *surfc* returnează un identificator de control al obiectului suprafaţă, care este copil al obiectului axe.

Exemplul 21.3.2. Reprezentaţi o suprafaţă 3D asociată cu liniile de nivel.
Cu secvenţa MATLAB:

```
[X,Y]=meshgrid(-3:.25:3);
Z=peaks(X,Y);
surfc(X,Y,Z)
```

se obţine reprezentarea grafică din figura 21.7.

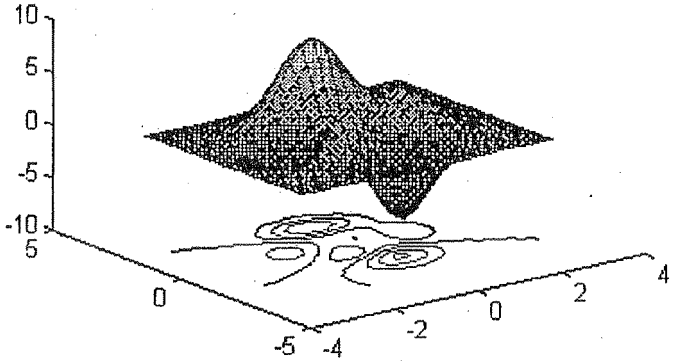


Fig. 21.7. Reprezentarea 3D a suprafeţelor asociate cu linii de nivel.

21.4. Reprezentarea obiectelor spaţiale

Obiectele spaţiale predefinite în MATLAB sunt:

cylinder Generează un obiect cilindru;
sphere Generează un obiect sferă.

21.4.1. Reprezentarea grafică a obiectului cilindru

Funcţia *cylinder* generează un cilindru de rază R, cu cercul bazei aproximat din N puncte echidistante; se apelează cu sintaxa:

[x,y,z]=*cylinder*(R,N)

Funcţia returnează matricele cu 2x(N+1) elemente, care specifică vârfurile fiecărei suprafeţe rezultate din aproximarea cercurilor bazelor cu poligoane cu N laturi. Vectorul R are două elemente [R1 R2], care precizează raza obiectului la partea inferioară (R1) şi superioară (R2), prin aceasta fiind posibilă construirea de conuri, trunchiuri de con, piramide, trunchiuri de piramide etc. Valoarea implicită este pentru R=[1 1], iar pentru N=20. Reprezentarea grafică se face cu funcţia *surface*(x,y,z). Omiterea argumentelor de ieşire determină reprezentarea grafică a obiectului cilindru.

Exemplul 21.4.1. Să se reprezinte grafic un con de rază Rc1=0.5 şi înălţime Hc=7.5 şi un trunchi de piramidă cu baza hexagon, cu cercurile circumscrise bazelor de raze Rp1=1 şi Rp2=0.5 şi înălţimea Hp=7.5.
Cu secvenţa MATLAB:

```
Rc1=0.5;  Rc2=0;  Hc=7.5; N=30;
xc,yc,zc]=cylinder([Rc1 Rc2],N);zc=zc*Hc;
view([-37.5 30]);surface(xc,yc,zc); grid
figure
Rp1=1;  Rp2=0.5;  Hp=7.5; N=6;
xp,yp,zp]=cylinder([Rp1 Rp2],N);zp=zp*Hp;
view([-37.5 30]);surface(xp,yp,zp); grid
```

se obţine reprezentarea grafică din figura 21.8.

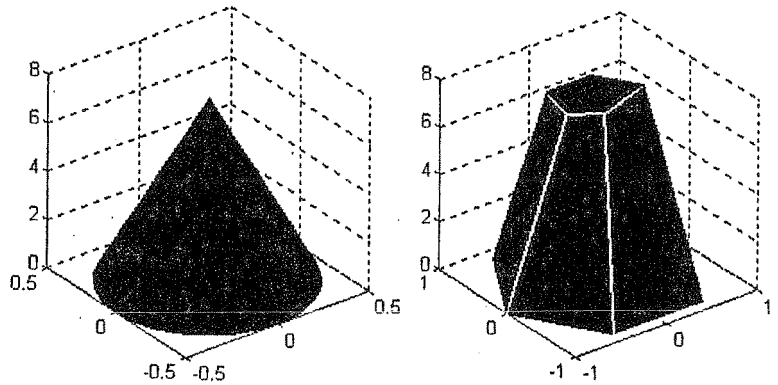


Fig. 21.8. Reprezentare grafică a obiectelor cu funcţia *cylinder*.

21.4.2. Reprezentarea grafică a obiectului sferă

Funcţia *sphere* generează coordonatele (x,y,z) ale sferei unitate, care pot fi utilizate cu funcţiile *surf* şi *mesh*; se apelează cu una dintre sintaxele:

[x,y,z]=*sphere*(n) - generează coordonatele sferei în trei matrice (n+1)×(n+1), care pot fi reprezentate grafic cu funcţia *surf*(x,y,z) sau *mesh*(x,y,z).
sphere(n) - reprezintă grafic suprafaţa unei sfere. Implicit n=20.

Exemplul 21.4.2. Să se genereze şi să se reprezinte grafic o sferă.
Cu secvenţa MATLAB :

```
[X,Y,Z]=sphere(20);
mesh(X,Y,Z);grid
```

se obţine reprezentarea grafică din figura 21.9.

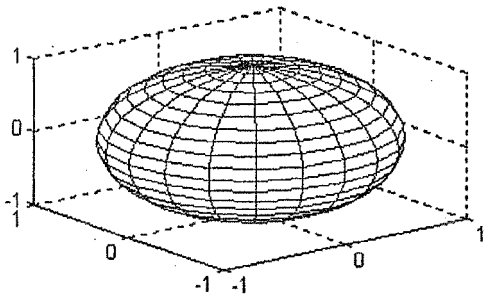


Fig. 21.9. Reprezentare grafică cu funcţia *sphere*.

21.5. Poziţionarea observatorului faţă de obiect

Poziţionarea observatorului poate fi precizată cu funcţiile:

view Specifică poziţia de vizualizare a unui grafic 3D;
hidden Permite vizualizarea sau ascunderea suprafeţelor suprapuse.

21.5.1. Definirea poziţiei observatorului faţă de obiect

Poziţia observatorului faţă de un obiect 3D se precizează prin unghiul pe orizontală (numit azimut) şi unghiul pe verticală (numit elevaţie). Funcţia *view* este utilizată pentru vizualizarea unei reprezentări grafice spaţiale din diverse poziţii; se apelează cu una dintre sintaxele :

view(az,elv), şi
view([az,elv]) - stabileşte unghiul din care se vede reprezentarea grafică 3D. *az* este azimutul (sau unghiul în plan orizontal), iar *elv* este unghiul pe verticală (ambele în grade). Azimutul poate lua valori pozitive sau negative, cea pozitivă fiind o rotire în jurul axei Oz în sens orar. Valorile pozitive ale unghiului pe verticală corespund unui punct de observare plasat deasupra planului (x,y), iar valorile negative corespund punctelor de sub plan;
view([x,y,z]) - stabileşte unghiul punctului de observare în coordonate carteziene;
[az,elv]=*view* - returnează unghiul pe orizontală şi unghiul pe verticală din care este observată o reprezentare grafică 3D;
view(2) - stabileşte reprezentarea grafică la 2D cu valorile simplificate: `az=0, elv=90`;
view(3) - stabileşte reprezentarea grafică 3D cu valorile implicite: `az=-37.5, elv=30`.

Pentru orientare:

elv=90 - vederea de deasupra obiectului;
az=elv=0 - vedere directă din planul zero;
az=180 - vedere din spate a obiectului.

21.5.2. Vizualizarea şi ascunderea suprafeţelor suprapuse

Funcţia *hidden* elimină liniile ascunse ale unei reţele de tip „mesh"; se apelează cu una dintre sintaxele:

hidden on - setează modul de reprezentare astfel încât liniile acoperite ale reţelei de tip „mesh" să fie ascunse. Acesta este modul implicit;
hidden off - setează modul de reprezentare astfel încât liniile acoperite ale reţelei de tip „mesh" să fie vizibile.

Funcţia *hidden* setează proprietatea `FaceColor` a obiectului suprafaţă. Liniile acoperite ascunse corespund la FaceColor =`BackgroundColor`, care în mod normal este negru (`black`), iar liniile acoperite vizibile corespund la `FaceColor=none`.

22. PERSONALIZAREA GRAFICELOR

Prin personalizarea graficelor se înţelege plasarea în câmpul acestora a unor texte, etichete ale axelor, a unităţilor de măsură, a scalelor, precum şi a titlului. Se utilizează în acest scop următoarele funcţii MATLAB:

title	Precizează titlul graficului;
xlabel	Precizează eticheta axei X;
ylabel	Precizează eticheta axei Y;
zlabel	Precizează eticheta axei Z;
text	Plasează un text pe grafic la o poziţie impusă;
gtext	Plasează un text pe grafic la poziţia selectabilă cu mouse-ul;
grid	Trasează o reţea de linii „grid" pe grafic.

22.1. Precizarea titlului graficului

Titlul graficului curent se plasează deasupra acestuia cu funcţia *title*; se apelează cu sintaxa:

title('text')

unde 'text' este un şir de caractere care reprezintă titlul graficului.
Instrucţiunea:

title(*date*)

plasează data în partea superioară a graficului curent.

Un exemplu de includere a variabilelor în titlu este prezentat de secvenţa:

```
a=2; t=4.5;  v=a*t;
title(['Viteza este ',num2str(v),' m/s'])
n=3;
title(['Cazul numarul ',int2str(n)])
```

22.2. Precizarea etichetelor axelor

Precizarea numelui mărimilor reprezentate pe fiecare axă, precum şi a unităţilor de măsură folosite, se face cu funcţiile *xlabel* (pentru axa x), *ylabel* (pentru axa y) şi *zlabel* (pentru axa z); se apelează cu sintaxele:

xlabel('text') *ylabel*('text') *zlabel*('text')

unde 'text' este un şir de caractere care reprezintă numele axei, unitatea de măsură etc. ale graficului curent. Repetarea instrucţiunii cu alt text conduce la înlocuirea textului anterior, fără a fi necesară refacerea reprezentării grafice.

22.3. Plasarea unui text pe grafic

Plasarea unui text în câmpul grafic se face cu funcţiile *text* sau *gtext*; se apelează cu sintaxele:

text(x,y,'text') *text*(x,y,z,'text')

Funcţia *text* scrie şirul de caractere „text" la coordonatele (x,y) sau (x,y,z) pe grafic. Coordonatele sunt date în unităţile de măsură ale ultimului grafic.

Dacă X şi Y sunt vectori, funcţia *text* scrie şirul de caractere „text" la toate poziţiile date de perechile (X,Y). Dacă „text" este un şir de aceeaşi lungime cu vectorii X şi Y, funcţia marchează fiecare punct cu linia, markerul sau caracterul corespunzător din şirul „text".

Apelată cu sintaxa *text*(X,Y,'text','sc'), funcţia interpretează punctele (X,Y) în coordonatele ecranului; (0.0,0.0) este colţul din stânga jos, iar (1.0,1.0) este colţul din dreapta sus al ecranului.

Plasarea unui text sau a unui comentariu în câmpul grafic cu mouse-ul utilizează funcţia *gtext*; se apelează cu sintaxa:

gtext('text')

Funcţia *gtext* afişează şirul de caractere 'text' în fereastra grafică şi aşteaptă deplasarea acestuia pe grafic cu mouse-ul. Apăsând un buton al mouse-ului sau oricare tastă, textul se scrie pe grafic la poziţia selectată.

22.4. Trasarea unei reţele („grid") pe grafic

Gridul este o reţea de linii verticale şi orizontale pe un grafic 2D sau 3D care uşurează citirea datelor. O astfel de reţea se trasează cu funcţia *grid* care se apelează cu sintaxa:

grid `on` *grid* `off`

Dacă funcţia *grid* este setată `on`, pe grafic sunt reprezentate liniile grid, iar dacă este `off`, acestea sunt eliminate. Funcţia *grid* stabileşte proprietăţile XGrid, YGrid şi ZGrid ale obiectului axe curente.

23. CREAREA ŞI CONTROLUL AXELOR

În acest capitol se explică următoarele funcţii MATLAB:

subplot Creează reprezentări grafice la poziţii fixe (împarte ecranul în mai multe ferestre grafice folosite pentru reperezentări multiple);
axes Creează axe la poziţii impuse;
gca Returnează caracteristicile axelor figurii curente;
cla Şterge axele curente;
axis Stabileşte sau returnează caracteristicile axelor;
hold Comandă menţinerea (îngheţarea) graficului curent pe ecran;
ishold Returnează starea funcţiei „hold".

23.1. Divizarea ferestrei grafice

Funcţia *subplot* creează şi controlează acoperirea ecranului cu ferestre grafice; se apelează cu una dintre sintaxele:

subplot(m,n,p) *subplot*(h)

Funcţia *subplot*(m,n,p) împarte ecranul într-o matrice m x n, creează axele în subfereastra p şi returnează identificatorul de control al noilor axe. Argumentele m, n şi p trebuie să fie numere întregi în intervalul [1, 9]. Scalarul m precizează în câte subferestre se împarte fereastra pe verticală, iar n specifică împărţirea pe orizontală. Dacă fereastra grafică a fost împărţită anterior, se face numai specificarea ca subfereastră grafică curentă.

Cu sintaxa *subplot*(h), unde h este un identificatorul de control al caracteristicilor axelor, se apelează o subfereastră în care urmează a se realiza o reprezentare grafică.

Comanda *clf* sau *subplot*(1,1,1) restaurează fereastra grafică iniţială, ştergând toate celelalte axe şi reprezentări.

Exemplul 23.1.1. Realizaţi reprezentarea funcţiei y=sin(x), atât în prima jumătate a ferestrei grafice, cât şi în ultima şesime a acesteia.
Cu secvenţa MATLAB:

```
y=sin(2*pi*(0:49)/20);
subplot(2,2,1); plot(y); grid
subplot(2,3,6); plot(y); grid
```

se obţine reprezentarea grafică din figura 23.1.

23.2. Schimbarea limitelor axelor

Funcţia *axis* care realizează scalarea axelor se apelează cu una dintre sintaxele:

axis([xmin xmax ymin ymax]) - stabileşte scalarea axelor unei reprezentări grafice 2D curente la valorile precizate de scalarii: xmin, xmax, ymin şi ymax;

axis([xmin xmax ymin ymax zmin zmax]) - stabileşte scalarea axelor unei reprezentări grafice 3D curente la valorile precizate de scalarii: xmin, xmax, ymin, ymax, zmin şi zmax;

axis('auto') - autoscalează reprezentarea grafică curentă, alegând cele mai bune limite, astfel încât să fie reprezentată integral;

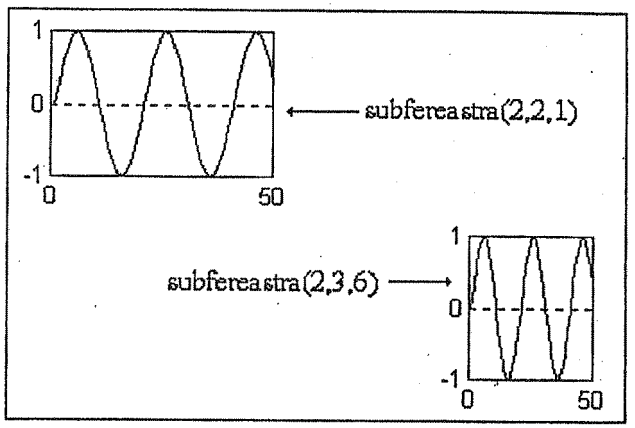


Fig. 23.1. Reprezentare în subferestrele grafice cu funcţia *subplot*.

axis(*axis*) - „îngheaţă" limitele axelor la valorile curente, astfel încât următoarele reprezentări să se facă cu aceleaşi limite;

a=*axis* - returnează un vector linie care conţine limitele reprezentării grafice curente. Vectorul are patru elemente pentru reprezentările grafice 2D şi şase elemente pentru reprezentările grafice 3D;

axis('ij') - realizează reprezentarea grafică în coordonatele matriceale. Originea sistemului de coordonate este colţul din stânga-sus. Axa i este verticală şi orientată de sus în jos, iar axa j este orizontală şi orientată de la stânga la dreapta;

axis('xy') - realizează reprezentarea grafică în coordonatele carteziene. Originea sistemului de coordonate este colţul din stânga-jos. Axa x este orizontală şi orientată de la stânga la dreapta, iar axa y este verticală şi orientată de jos în sus;

axis('square') - stabileşte axele curente la o formă pătrată;

axis('equal') - realizează reprezentarea grafică cu aceeaşi scală pe toate axele;

axis('on') - reface etichetele axelor graficelor;

axis('off') - şterge etichetele axelor graficelor;

[s1,s2,s3]=*axis*('state') - returnează trei şiruri de caractere care indică starea curentă a proprietăţilor etichetelor axelor, după cum urmează:

s1 = 'auto' sau 'manual'
s2 = 'on' sau 'off'
s3 = 'xy' sau 'ij'

Fără argumente de ieşire, comanda *axis*(s1, s2, s3) reface proprietăţile axelor la valorile indicate de cele trei şiruri de caractere. Valoarea implicită este:

axis('auto', 'on', 'xy')

23.3. Suprapunerea succesivă a graficelor

Funcţia *hold* „memorează" graficul curent şi adaugă următoarele reprezentări grafice peste acesta; se apelează cu una dintre sintaxele:

`hold on` - „memorează" graficul curent şi toate proprietăţile axelor, adăugând următoarele grafice peste cel curent;

`hold off` - returnează starea iniţială prin care fiecare comandă *plot* şterge graficul anterior şi proprietăţile axelor înainte de a desena unul nou.

Dacă în fereastra grafică există mai multe subferestre grafice, fiecare dintre acestea are o stare a funcţiei *hold*, stabilită ca proprietate a obiectelor figură sau axe.

Pentru a testa starea funcţiei *hold* se foloseşte funcţia *ishold*, care returnează 1 dacă *hold* este activă (suprapune reprezentările grafice) şi 0 dacă este dezactivată (fiecare nouă reprezentare o şterge pe cea anterioară); se apelează cu sintaxa:

a=*ishold*

Dacă funcţia *hold* este activă, toate reprezentările grafice ulterioare se fac cu aceleaşi proprietăţi ale axelor. În acest caz, proprietatea `NextPlot` a ambelor obiecte, figură şi axe, este stabilită la „`add`".

23.4. Ştergerea axelor curente

Funcţia *cla* şterge axele curente; se apelează cu una dintre sintaxele:

cla - şterge toate obiectele (linii, texte, module obiect, suprafeţe şi imagini) din axele curente;

cla reset - iniţializează la valorile implicite toate obiectele din axele curente, precum şi toate proprietăţile axelor, cu excepţia poziţiei.

23.5. Determinarea identificatorului axelor curente

Funcţia *gca* returnează identificatorul de control al caracteristicilor axelor curente; se apelează cu sintaxa:

h=*gca*

Prin axe curente se înţeleg axele în care funcţiile *plot*, *title*, *surf*, *mesh* etc. reprezintă grafic rezultatele. Fiecare figură are o axă curentă. Orice schimbare în figura curentă determină funcţia *gca* să returneze identificatorul de control al noii figuri curente. Pentru a schimba axele curente sau pentru a crea altele noi, se folosesc funcţiile *axes* sau *subplot*. Identificatorul de control se foloseşte apoi asociat cu funcţiile *get* sau *set* pentru a vedea sau modifica proprietăţile obiectelor axe.

23.6. Crearea unui obiect grafic axe

Funcţia *axes* creează axele la poziţia specificată; se apelează cu una dintre sintaxele :

h=*axes* *axes*(h)

h=*axes* ('Nume_proprietate', Valoare_ proprietate, ...)

h=*axes* - creează axe pe întreaga fereastră şi returnează identificatorul de control h al caracteristicilor obiectului axe;

axes(h) - creează un obiect axe curent cu identificatorul de control h.

Funcţia *axes* este o funcţie de nivel scăzut pentru crearea obiectului axe. Obiectele axe (*axes*) sunt copii (children) ai obiectelor figură (*figure*) şi părinţi (parent) ai obiectelor imagine (*image*), linie (*line*), suprafaţă (*surface*) şi text (*text*). Obiectele axe definesc cadrul de referinţă în interiorul căruia sunt orientate obiectele copil.

Funcţia *axes* creează obiecte grafice axe şi acceptă perechile nume/valoare_proprietate ca argumente de intrare. Aceste proprietăţi, care controlează diferite aspecte ale obiectului axe, sunt descrise în continuare şi pot fi stabilite sau citite utilizând funcţiile *set* şi *get*.

Funcţia *axis* (nu *axes*) permite un acces uşor la cele mai utilizate proprietăţi care controlează scalarea şi caracteristicile axelor.

Pentru obţinerea identificatorului axelor curente se utilizează funcţia *gca* (get current axes). Prin axe curente se înţeleg axele reprezentării grafice din fereastra activă.

O proprietate importantă a obiectului axe este poziţia (`Position`). Aceasta defineşte localizarea axelor în interiorul ferestrei grafice. Spre exemplu :

h=*axes* ('`Position`', poz)

creează un obiect axe care specifică poziţia în interiorul figurii curente din fereastra grafică şi returnează identificatorul de control h. Localizarea şi dimensiunea axelor este specificată de vectorul :

poz = [oriz vert lungimea lăţimea]

unde :

- oriz şi vert - sunt distanţele, pe orizontală şi pe verticală, de la colţul din stânga-jos al figurii până la colţul din stânga-jos al axelor de coordonate;
- lungimea şi lăţimea - sunt dimensiunile axelor.

Elementele vectorului poz sunt date în unităţile de măsură specificate de proprietatea `Units`. MATLAB-ul utilizează implicit unităţile normalizate, la care (0,0) este colţul din stânga-jos, iar (1,1) este colţul din dreapta-sus al ferestrei grafice (figura 23.2).

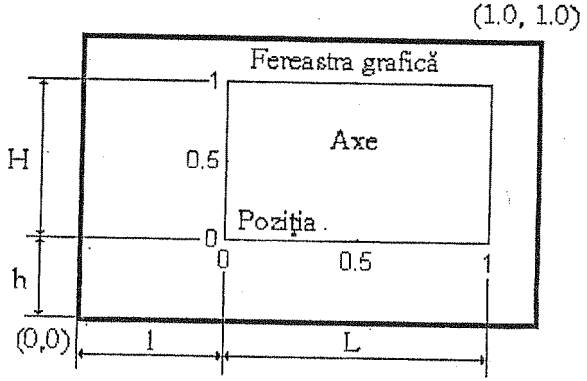


Fig. 23.2. Definirea parametrilor vectorului `Position`.

Exemplul 23.6.1. Să se creeze o reprezentare grafică care să aibă două obiecte grafice de tip axe, primul grafic cu originea la (0.1, 0.1) şi dimensiunile axelor 0.8 şi 0.6 din dimensiunea ferestrei grafice, iar al doilea grafic cu originea la (0.1, 0.8) şi dimensiunile axelor 0.8 şi 0.1.
Cu secvenţa MATLAB:

```
clf
axes ('Position',[.1 .1 .8 .6])
mesh (peaks)
axes ('Position',[.1 .8 .8 .1])
pcolor ([1:10; 1:10])
```

se obţine reprezentarea grafică din figura 23.3.

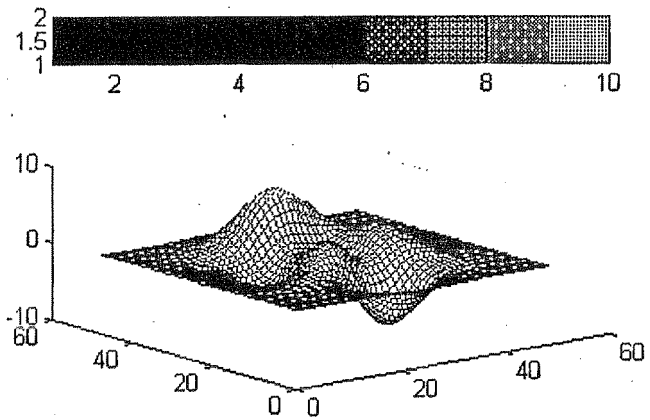


Fig. 23.3. Plasarea mai multor axe în fereastra grafică.

23.6.1. Proprietăţile generale ale obiectului grafic axe

Proprietăţile obiectului axe pot fi specificate la crearea acestora prin includerea perechilor nume/valoare_proprietate, sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `AspectRatio` [raport_axe, raport_date]

Această proprietate controlează raportul celor două axe la reprezentările 2D, care specifică:

- raportul dintre lungimea axelor verticale şi lungimea axelor orizontale (raport_axe);
- raportul dintre lungimea unităţii axei verticale şi lungimea unităţii axei orizontale (raport_date).

MATLAB-ul utilizează raportul specificat în „raport_axe", pentru care se umple cel mai bine dreptunghiul definit prin proprietatea `Position`. La crearea axelor cu „raport_date" specificat, MATLAB-ul schimbă limitele unei axe şi menţine raportul

lungimilor specificat în „raport_axe". Schimbările limitelor axelor nu afectează proprietatea limitelor (`Xlim` sau `Ylim`). Aceste raporturi sunt numere în intervalul [0, ∞). Ambele rapoarte sunt implicit NaN (rapoarte nespecificate). La prima folosire calculatorul alege cele mai bune rapoarte care realizează o reprezentare corespunzător încadrată în fereastra grafică.

2. `Box` on | off

Această proprietate specifică dacă graficul este încadrat de un dreptunghi pentru reprezentări 2D sau de un cub pentru reprezentări 3D.

3. `Children` vector al identificatorilor

Această proprietate este un vector al identificatorilor obiectelor reprezentate în axe. Obiectele copil ale axelor pot fi imagini, linii, suprafeţe şi text.

4. `Clim` [Cmin, Cmax]

Această proprietate determină modalitatea după care se transformă valorile datelor în matricea de culoare. MATLAB transformă implicit valoarea minimă în prima culoare a matricei de culoare şi valoarea maximă în ultima culoare a matricei de culoare. În acest caz, întregul interval al matricei de culoare este asociat întregului interval al datelor de intrare. Există posibilitatea schimbării acestei transformări prin setarea lui Cmin la valoarea primei culori dorite din tabela culorilor şi a lui Cmax la ultima culoare dorită din aceeaşi tabelă. În acest caz, datele de intrare mai mici decât Cmin şi mai mari decât Cmax sunt omise, devenind transparente. Uneori însă, este utilă reprezentarea numai într-o zonă a intervalului de culoare, rezervând pentru alte reprezentări un anumit interval. Prin urmare, se foloseşte o porţiune limitată a matricei de culoare. Această proprietate afectează obiectele suprafaţă, însă nu şi imaginile, liniile şi textul.

5. `ClimMode` auto | manual

Dacă `ClimMode` este setat „auto", MATLAB calculează limitele axelor de culoare pe care le transmite în întregime axelor copil. Dacă `ClimMode` este setat „manual", limitele axelor de culoare se schimbă prin setarea proprietăţii `Clim` la valoarea dorită.

6. `Color` Culoarea specificată | non

Această proprietate specifică culoarea folosită pentru a umple aria definită prin proprietatea `Position` (fondul reprezentării grafice din axele curente sau din axele specificate de identificatorul h). Culoarea poate fi specificată utilizând un vector al valorilor RGB sau un nume de culoare predefinită în MATLAB. Valoarea implicită a acestei culori este cea a culorii de fond a ferestrei grafice (background).

7. `ColorOrder` matrice m x 3 sau valori RGB.

Această proprietate defineşte ordinea culorilor folosite de reprezentările grafice, fiind o matrice m x 3 cu valorile RGB. Dacă culorile liniilor reprezentărilor grafice realizate cu funcţiile *plot* şi *plot3* nu sunt specificate, acestea aleg ciclic culorile predefinite, în ordinea : galben-mov-albastru_ marin - roşu - verde - albastru.

8. `GridLineStyle` tip_linie

Această proprietate definește tipul de linie folosită pentru desenarea liniilor grid. Sunt acceptate următoarele tipuri de linii (cu simbolurile precizate în paranteze): linie continuă (-); linie întreruptă (--) ; linie-punct (-.) ; linie din puncte (:).

Specificarea liniei dorite se face cu comanda:

set(gca,'`GridLineStyle`','tip_linie')

9. `NextPlot` new | add | replace

Această proprietate este folosită de toate funcțiile care utilizează obiectul axe: *plot, plot3, fill, fill3, mesh, surf, bar* etc.

Opțiunea „new" crează noi axe înainte de a face o reprezentare grafică; opțiunea „add" adaugă noi obiecte la axele curente, iar „replace" (opțiune implicită) șterge axele curente și creează altele noi în aceeași poziție. Dacă funcția *hold* este setată „off", proprietatea `NextPlot` este stabilită la opțiunea „replace", iar dacă *hold* este „on", `NextPlot` este setat pe „add".

10. `Parent` identificator (numai de citit)

Această proprietate returnează identificatorul obiectului axe, care este părinte al obiectului figură. Funcția *gca* returnează, de asemenea, identificatorul axelor curente ale figurii.

11. `Position` vector [oriz vert lungimea lățimea]

Această proprietate specifică mărimea și localizarea axelor în fereastra grafică definită prin vectorul:

poz = [oriz vert lungimea lățimea]

unde :

- oriz și vert - sunt distanțele, pe orizontală și pe verticală, de la colțul din stânga-jos al figurii până la colțul din stânga-jos al axelor de coordonate;
- lungimea și lățimea - sunt dimensiunile axelor.

Elementele vectorului poz sunt date în unitățile de măsură specificate de proprietatea `Units`. MATLAB-ul utilizează implicit unitățile normalizate, la care (0,0) este colțul din stânga-jos, iar (1,1) este colțul din dreapta-sus al ferestrei grafice.

12. `TickLength` vector [2D 3D]

Această proprietate specifică lungimea liniuței care divizează axele, ca raport al dimensiunilor axelor. Proprietatea este descrisă de un vector cu două elemente, primul pentru reprezentările 2D, iar al doilea pentru reprezentările 3D. Valorile implicite sunt [0.01 .0025].

13. `TickDir` in | out

Această proprietate specifică dacă liniile care marchează diviziunile axei sunt orientate spre interior (in) sau spre exterior (out).

14. `Type` Șir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul axe, șirul de caractere este întotdeauna *„axes"*.

15. `Units` pixels | normalized | inches | centimeters | points

Această proprietate specifică unitățile de măsură utilizate pentru interpretarea proprietății `Position`. În toate cazurile măsurarea se face din colțul stânga-jos al ferestrei grafice. Unitățile normalizate transformă colțul din stânga-jos al ferestrei grafice în coordonatele (0,0), iar colțul din dreapta-sus în (1.0, 1.0). Celelalte unități de măsură sunt unități absolute, iar unitatea punct (points) este egală cu 1/72 inch.

16. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului, care poate să nu fie folosită, dar poate fi văzută folosind funcția *get*.

17. `View` [az, elv]

Această proprietate stabilește punctul de observare al reprezentării grafice 3D, caracterizat prin azimut (unghiul de rotire în plan orizontal) și elevație (unghiul cu planul orizontal). Valorile pozitive ale azimutului au semnificația unei rotiri în sensul acelor de ceasornic în jurul axei z. Valorile pozitive ale elevației permit o vedere de deasupra a obiectivului, iar valorile negative o vedere de sub obiect. Ambele unghiuri sunt date în grade.

18. `Visible` on | off

Această proprietate determină dacă obiectul este sau nu vizibil pe ecran. Dacă proprietatea `Visible` este stabilită pe „off", liniile axelor, marcajele de divizare și etichetele nu sunt vizibile. Copiii obiectelor axe nu sunt afectați de proprietatea `Visible` a părinților.

23.6.2. Proprietățile de control ale axelor (X, Y și Z)

În acest subcapitol se prezintă proprietățile de control ale axelor (X, Y și Z). Deși sunt detaliate numai pentru axa X, acestea se aplică identic la axele Y și Z prin înlocuirea primului caracter, X, cu Y, respectiv Z.

1. `XColor` Culorile specificate

Această proprietate setează culoarea axei X, a liniilor de divizare, a etichetelor care marchează diviziunile și a liniilor grid ale acesteia.

2. `XDir` normal | reverse

Această proprietate stabilește direcția de creștere a valorilor numerice pe axa X, de la stânga la dreapta (normal) sau invers (reverse).

3. `XGrid` on | off

Proprietatea `XGrid` stabilită pe „on" reprezintă linii grid perpendiculare pe axa X în dreptul fiecărei diviziuni a acesteia.

4. XLabel text (identificator)

Această proprietate este identificator al obiectului text utilizat pentru etichetarea axei X. Pentru a avea un identificator trebuie creat un obiect text, ca în exemplul:

```
set(gca,'xlabel',text(0.5,0,'Axa X'))
```

O metodă simplă de plasare a etichetei axei X este funcția *xlabel*.

5. `XLim` [Xmin Xmax]

Această proprietate precizează valorile minimă şi maximă ale axei X. Aceste valori pot fi pozitive sau negative.

6. `XLimMode` auto | manual

Dacă proprietatea `XLimMode` este stabilită pe „auto", reprezentarea se face cu autoscalare; în caz contrar se reprezintă zonele selectate „manual".

7. `XScale` linear | log

Această proprietate specifică modul de scalare (liniar sau logaritmic) al axei X.

8. `XTick` vector

Această proprietate specifică localizarea şi valoarea diviziunilor pe axa X. În exemplul:

```
set(gca,'XTick',[0 .25 .5 1 2])
```

se stabilesc ca diviziuni ale axei X elementele: 0, 0.25, 0.5, 1, 2.

9. `XTickMode` auto | manual

Dacă proprietatea XTickMode este stabilită pe „auto" , distanţa dintre diviziuni este calculată automat; în caz contrar acestea trebuie precizate „manual".

10. `XTickLabel` matrice şir de caractere

Această proprietate specifică textul care se utilizează ca etichete pentru axa X. Aceste etichete înlocuiesc etichetele numerice generate de MATLAB, ca în exemplul :

```
set(gca,'XTickLabels',['Data 1';'Data 2'])
```


24. CREAREA ŞI CONTROLUL FIGURILOR

Crearea şi controlul figurilor se face cu următoarele funcţii MATLAB:

figure Creează o figură lăsând neşterse figurile anterioare;
gcf Returnează caracteristicile figurii curente;
clf Şterge figura curentă;
close Închide figura curentă.

24.1. Crearea unui obiect figură

Funcţia *figure* deschide o nouă fereastră grafică destinată creării unui obiect figură; se apelează cu una dintre sintaxele:

h= *figure* *figure*(h)
h= *figure* ('Nume_proprietate', Valoare_ proprietate, ...)

Obiectele figură sunt copii ai obiectului grafic rădăcină şi părinţi ai obiectelor axe (*axes*), elemetelor de control ale interfeţelor grafice (*uicontrol*) şi meniurilor (*uimenu*).

h=*figure* - deschide o nouă fereastră grafică şi returnează numărul acesteia, cunoscut sub numele de identificator al figurii. Aceşti identificatori sunt numere întregi, care se afişează în partea superioară a ferestrei grafice ca număr al figurii.

figure(h) - prezintă reprezentarea grafică cu identificatorul h;

Funcţia *figure* creează obiecte grafice figură şi acceptă perechile nume / valoare_proprietate, ca argumente de intrare. Aceste proprietăţi, care controlează diferite aspecte ale obiectului figură, sunt descrise la subcapitolul „Proprietăţile obiectului figură" şi pot fi stabilite sau citite, utilizând funcţiile *set* şi *get*.

Pentru obţinerea identificatorului axelor curente se utilizează funcţia *gcf* (get current figure). Prin figură curentă se înţelege fereastra grafică în care este reprezentată oricare dintre comenzile grafice MATLAB.

24.2. Determinarea identificatorului unei figuri

Funcţia *gcf* returnează identificatorul de control al caracteristicilor figurii curente; se apelează cu sintaxa:

h=*gcf*

Prin figură curentă se înţelege reprezentarea grafică realizată cu funcţiile *plot*, *title*, *surf*, *mesh*, etc. Fiecare figură are o axă curentă. Pentru a schimba figura curentă se foloseşte funcţia *figure*. Identificatorul de control se foloseşte apoi asociat cu funcţiile *get* sau *set* pentru a vedea sau modifica proprietăţile obiectelor figuri.

24.3. Proprietăţile obiectului figură

Proprietăţile obiectului figură pot fi specificate la crearea acestuia prin includerea perechilor nume/valoare_proprietate sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `Children` vector al identificatorilor

Această proprietate este un vector al identificatorilor obiectelor reprezentate în figură. Obiectele copil ale figurii sunt: axele, elementele de control ale interfeţelor grafice şi meniurile.

2. `Color` Culoarea specificată

Această proprietate defineşte culoarea folosită pentru fondul ferestrei grafice a figurii. Culoarea poate fi specificată utilizând un vector al valorilor RGB sau un nume de culoare predefinită în MATLAB. Valoarea implicită a acestei culori este cea a culorii de fond a ferestrei grafice (background).

3. `Colormap` matrice m x 3 a valorilor RGB

Această proprietate este o matrice m x 3 a ponderii culorilor fundamentale (RGB) care definesc o culoare individuală. Comenzile grafice ataşează fiecărui triplet câte o valoare, culorile fiind apelate prin indecşii corespunzători. Spre exemplu, indexul 1 specifică primul triplet RGB, indexul 2 specifică al doilea triplet RGB etc. Matricea de culoare poate avea orice lungime, însă trebuie să aibă trei coloane. Matricea de culoare a figurii implicite are 64 de culori predefinite. Aceasta poate fi înlocuită prin orice altă matrice definită de utilizator sau dintre cele predefinite. Pentru definirea matricei de culoare trebuie specificată intensitatea culorii roşu în prima coloană, a culorii verde în a doua coloană şi a culorii albastru în a treia coloană. Intensitatea culorilor este un număr real cuprins între 0.0 şi 1.0. Valoarea 0 indică lipsa componentei respective, iar valoarea 1 indică intensitatea maximă a acesteia. Astfel, tripletul RGB [0,0,0] reprezintă culoarea negru, iar [1, 1, 1] reprezintă culoarea alb.

4. `CurrentAxes` identificatorul axelor

Această proprietate returnează identificatorul axelor curente ale figurii (fiind identic cu identificatorul obţinut prin funcţia *gca*). În toate figurile care au axe copii acest identificator este acela al axelor curente. Axele pot fi făcute curente folosind funcţiile *axes* şi *set*. Dacă nu există axe copil într-o figură, se pot crea unele cu comanda: h=get(gcf,'CurrentAxes'), care returnează şi identificatorul h.

5. `CurrentMenu` identificator al meniului

Această proprietate conţine identificatorul celui mai recent meniu selectat. Dacă se specifică un singur identificator pentru meniuri cu mai mult de o opţiune, este necesară o apelare a acestei proprietăţi pentru a vedea care meniu a fost apelat.

6. `CurrentObject` identificator

Această proprietate returnează identificatorul obiectului care se găseşte la punctul curent (proprietate `CurrentPoint`). În cazul obiectelor multiple suprapuse, obiectul selectat este cel care a fost primul plasat în figură. Se poate utiliza această proprietate pentru a determina ordinea în care au fost plasate obiectele grafice într-o figură.

7. `CurrentCharacter` caracter (numai pentru citit)

Această proprietate returnează ultima tastă apăsată în figura curentă, care se poate obţine cu instrucţiunea: get(gcf,'CurrentCharacter').

8. `CurrentPoint` [x y]

Această proprietate returnează coordonatele x şi y ale poziţiei în care a fost apăsat sau eliberat butonul mouse-ului. Aceste coordonate se selectează prin apăsarea sau eliberarea oricărui buton al mouse-ului în fereastra grafică. Coordonatele punctului curent sunt măsurate din colţul stânga-jos al ferestrei grafice, în unităţi precizate de proprietatea Units. Proprietatea se poate folosi şi pentru a urmări locul şi numărul de puncte selectate cu mouse-ul în fereastra grafică.

9. `InvertHardcopy` on | off

Această proprietate are efect numai la tipărirea ferestrei grafice. Deoarece implicit fondul ferestrei grafice a MATLAB-ului este de culoare neagră, pentru a tipări la imprimantă pe un fond alb, se foloşte proprietatea `InvertHardcopy`. Dacă `InvertHardcopy` este „on", MATLAB converteşte fondul reprezentării grafice la culoarea alb şi axele, subdiviziunile, etichetele axelor etc., la culoarea negru. Această proprietate nu are efect asupra culorilor folosite pentru celelalte obiecte grafice din reprezentare. Dacă `InvertHardcopy` este „off", se tipăreşte fereastra grafică aşa cum este pe ecran.

10. `MenuBar` none | figure

Această proprietate permite controlul în bara superioară de meniuri a ferestrei grafice. Dacă este stabilită pe „figure", la partea superioară a ferestrei grafice se găseşte bara de meniuri, iar dacă este stabilită pe „none", bara de meniuri lipseşte.

11. `MinColormap` scalar (implicit = 64)

Această proprietate stabileşte numărul de culori folosite de matricea de culoare cu instrucţiunea: `h=set(gcf,'MinColorMap',256)`.

12. `Name` şir de caractere

Această proprietate permite afişarea şirului de caractere menţionat în bara de titlu a ferestrei grafice. Implicit, acest şir este gol, fiind afişat numai: `Figure No.1`, `Figure No.2` etc. Dacă se precizează acest şir, spre exemplu „ANALIZA DATELOR", atunci se va afişaţă: `Figure No.1`:„ANALIZA DATELOR".

13. `NextPlot` new | add | replace]

Această proprietate este utilizată de funcţiile grafice de nivel scăzut: `plot`, `plot3`, `fill`, `fill3`, precum şi de funcţiile grafice-M: `mesh`, `surf`, `bar` etc., şi precizează modul cum se construieşte următorul grafic. Opţiunea `new` crează o fereastră grafică nouă înainte de a afişa reprezentarea grafică. Opţiunea `add` (implicită) utilizează pentru reprezentarea grafică figura curentă, iar opţiunea `replace` şterge toate proprietăţile figurii anterioare, precum şi toţi copiii figurii, cu excepţia poziţiei (`Position`). Comanda *subplot* utilizează această proprietate; subplot(1,1,1) stabileşte `NextPlot` la `replace`.

14. `NumberTitle` on | off

Această proprietate precizează dacă şirul `Figure No.N` (unde N este numărul figurii), este afişat în bara superioară a titlului ferestrei grafice („on") sau („off").

15. `PaperOrientation` portrait | landscape

Această proprietate precizează orientarea figurii pe pagină la tipărire. Orientarea `portrait` este cu dimensiunea cea mai mare a paginii pe verticală, iar `landscape` este cu dimensiunea cea mai mare pe orizontală.

16. `PaperPosition` vector linie cu 4 elemente

Această proprietate precizează localizarea dreptunghiului în care se tipăreşte figura pe pagină. Acest dreptunghi se defineşte prin:

poz=[orizontal, vertical, lungimea, lăţimea]

unde :

- orizontal şi vertical sunt distanţele, pe orizontală şi verticală, de la colţul din stânga-jos al paginii până la colţul din stânga-jos al dreptunghiului care defineşte figura;
- lungimea şi lăţimea sunt dimensiunile dreptunghiului care defineşte figura.

Elementele vectorului poz sunt date în unităţile de măsură specificate de proprietatea `PaperUnits`.

17. `PaperSize` [lungimea lăţimea] (numai pentru citire)

Această proprietate conţine dimensiunea tipului de pagină (`PaperType`), măsurată în unităţile specificate de `PaperUnits`.

18. `PaperType` usletter | uslegal | a4letter

Această proprietate este folosită pentru a scala figura trimisă la imprimantă .

19. `PaperUnits` inches | centimeters | normalized | points

Această proprietate specifică unităţile de măsură folosite pentru definirea poziţiei figurii pe pagină (`PaperPosition`) şi a dimensiunii paginii (`PaperSize`). În toate cazurile măsurarea se face din colţul stânga-jos al paginii. Unităţile normalizate transformă colţul din stânga-jos al paginii în coordonatele (0,0), iar colţul din stânga-sus în (1.0, 1.0). Unitatea punct (points) este egală cu 1/72 inch.

20. `Parent` identificator (numai de citit)

Această proprietate returnează identificatorul obiectului rădăcină, care este părinte al obiectului figură. Identificatorul obiectului rădăcină este întotdeauna 0.

21. `Pointer` crosshair | arrow | watch | topl | topr | botl | botr | circle | cross | fleur

Această proprietate permite stabilirea semnului grafic folosit drept cursor pentru a indica poziţia mouse-ul în fereastra grafică. Sunt posibile următoarele cursoare: cruce (croshair), săgeată (arrow), clepsidră (watch), săgeată dublă înclinată dreapta (topl sau botr), săgeată dublă înclinată stânga (topr sau botl), cerc (circle), cruce-dublă (cross) sau cruce cu săgeţi (fleur).

22. `Position` vector linie cu 4 elemente

Această proprietate specifică mărimea şi localizarea ferestrei grafice pe ecran, definită prin vectorul:

poz = [orizontal vertical lungimea lăţimea]

unde :

- orizontal şi vertical sunt distanţele, pe orizontală şi pe verticală, de la colţul din stânga-jos al ecranului până la colţul din stânga-jos al ferestrei grafice;
- lungimea şi lăţimea sunt dimensiunile ferestrei grafice.

Elementele vectorului poz sunt date în unităţile de măsură specificate de proprietatea `Units`. Pentru a determina poziţia ferestrei grafice curente se poate utiliza funcţia *get*, iar pentru a muta fereastra în altă poziţie se foloseşte funcţia *set*:

```
get(gcf,'Position')
```

şi,

```
set(gcf,'Position',[100 200 225 150])
```

23. `Resize` on | off

Dacă această proprietate este stabilită pe „on", dimensiunea ferestrei grafice se poate modifica cu mouse-ul; în caz contrar nu se poate modifica.

24. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul figură şirul de caractere este întotdeauna *„figure"*.

25. `Units` pixels | normalized | inches | centimeters | points

Această proprietate specifică unităţile de măsură utilizate pentru interpretarea dimensiunilor şi localizarea datelor. În toate cazurile măsurarea se face din colţul stânga-jos al ferestrei grafice. Unităţile normalizate transformă colţul din stânga-jos al ferestrei grafice în coordonatele (0,0), iar colţul din dreapta-sus în (1.0, 1.0). Celelalte unităţi de măsură sunt unităţi absolute, iar unitatea punct (points) este egală cu 1/72 inch.

Această proprietate afectează proprietăţile `CurentPoint` şi `Position`. Specificarea perechilor nume/valoare_proprietate la crearea unui obiect trebuie făcută înaintea specificării proprietăţilor care utilizează aceste unităţi.

26. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului; poate fi afişată folosind funcţia *get*.

27. `Visible` on | off

Această proprietate determină dacă obiectul este sau nu vizibil pe ecran. Dacă proprietatea `Visible` este stabilită pe „off", fereastra grafică nu este afişată.

24.4. Ştergerea unei figuri

Funcţia *clf* şterge fereastra grafică sau figura curentă; se apelează cu una dintre sintaxele:

clf - şterge toate obiectele (axele) din figura curentă;

clf reset - restabileşte toate obiectele din axele curente, precum şi toate proprietăţile axelor, cu excepţia poziţiei, la valorile implicite;

24.5. Închiderea unei figuri

Funcţia *close* închide fereastra grafică necondiţionat şi fără confirmare; se apelează cu una dintre sintaxele:

close(h) *close*

Apelată fără argument de intrare, funcţia *close* închide fereastra grafică curentă; în caz contrar, închide fereastra cu caracterul de control h.

25. CREAREA UNOR OBIECTE GRAFICE

Funcţiile MATLAB pentru crearea obiectelor grafice sunt:

text Creează obiectul grafic text;
line Creează obiectul grafic linie;
surface Creează obiectul grafic suprafaţă;
image Creează obiectul grafic imagine.

25.1. Crearea unui obiect grafic text

Pentru crearea unui obiect text, care se poate adăuga la o reprezentare grafică, se foloseşte funcţia *text*, se apelează cu una dintre sintaxele:

h= *text*(x,y,'text') h= *text*(x,y,z,'text')
h= *text*('Nume_proprietate', Valoare_ proprietate, ...)

text(x,y,'text') - adaugă şirul de caractere dintre apostrofuri ('text') în poziţia specificată de coordonatele (x,y) ale axelor curente, date în unităţile acestora;

text(x,y,z,'text') - adaugă un text în poziţia precizată de coordonatele 3D;

h=*text* - returnează un vector coloană al identificatorilor obiectelor text, câte unul pentru fiecare obiect text.

Funcţia *text* este de nivel înalt, atunci când este folosită direct pentru a adăuga texte pe un grafic şi este de nivel scăzut, atunci când creează obiecte text.

Obiectele text sunt copii ai obiectelor axe. Funcţia *text* creează obiecte grafice text şi acceptă perechile nume / valoare_proprietate, ca argumente de intrare. Aceste proprietăţi, care controlează diferite aspecte ale obiectului figură, sunt descrise mai jos şi pot fi setate sau citite, utilizând funcţiile *set* şi *get*.

Perechile (x,y), sau (x,y,z) pot fi urmate de perechile nume / valoare_proprietate pentru a specifica proprietăţi suplimentare ale obiectului text. De asemenea, coordonatele pot fi omise, precizându-se numai perechile nume / valoare_proprietate, pentru modificarea unui obiect creat anterior.

25.1.1. Proprietăţile obiectului grafic text

Proprietăţile obiectului text pot fi specificate la crearea acestuia prin includerea perechilor nume/valoare_proprietate, sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `Children` vector al identificatorilor

Obiectul text nu are copii, atfel încât această proprietate este întotdeauna o matrice goală.

2. `Clipping` on | off

Dacă proprietatea `Clipping` este stabilită pe „on", orice porţiune de text din afara zonei definită de dreptunghiul axelor nu va fi afişată. Valoarea implicită este „off".

3. `Color` Culoarea specificată

Această proprietate precizează culoarea folosită pentru text. Culoarea poate fi specificată utilizând un vector al valorilor RGB, sau un nume de culoare predefinită în MATLAB. Valoarea implicită a culorii este alb.

4. `Extent` vector linie cu 4 elemente

Această proprietate precizează poziţia şi dimensiunile dreptunghiului care încadrează şirul de caractere (textul). Specificarea acestuia se face cu un vector de forma:

poz=[orizontal, vertical, lungimea, lăţimea]

unde :

- orizontal şi vertical sunt distanţele, pe orizontală şi verticală, de la colţul din stânga-jos al dreptunghiului care defineşte figura până la colţul din stânga-jos al dreptunghiului care încadrează textul;
- lungimea şi lăţimea sunt dimensiunile dreptunghiului care încadrează textul.

Elementele vectorului poz sunt date în unităţile de măsură specificate de proprietatea `Units`.

5. `FontAngle` normal | italic | oblique

Această proprietate specifică modul de scriere: drept (normal), italic sau înclinat.

6. `FontName` şir de caractere

Această proprietate stabileşte numele de fonturi folosite de obiectul text. Numele fonturilor este unul dintre cele gestionate de **Windows**, şi care se găsesc în subdirectorul **System**.

7. `FontSize` scalar

Această proprietate stabileşte dimensiunea fonturilor folosite de obiectul text.

8. `FontStrikeThrough` on | off

Dacă această proprietate este stabilită pe „on", textul este afişat tăiat cu o linie, iar dacă este „off" textul apare normal.

9. `FontUnderline` on | off

Dacă această proprietate este stabilită pe „off", textul afişat este nesubliniat, iar dacă este setată „on" este subliniat.

10. `FontWeight` light | normal | demi | bold

Această proprietate precizează grosimea liniilor folosite pentru scrierea textului: subţire (light), normală, semibold (demi) şi bold.

11. `HorizontalAlignment` left | center | right

Această proprietate specifică tipul de aliniere a textului pe orizontală faţă de punctul menţionat de proprietatea `Position`. Alinierea poate fi: la stânga (left), centrat (center) sau la dreapta (right).

12. `Parent` identificator (numai de citit)

Această proprietate returnează identificatorul obiectului axe, care este părinte al obiectului text.

13. `Position` vector cu două (trei) elemente

Această proprietate precizează poziţia textului în 3D (dacă al treilea element este omis, el este considerat zero). Unităţile folosite sunt cele precizate pentru proprietatea `Units`.

14. `Rotation` scalar

Această proprietate stabileşte orientarea textului la una din cele şapte posibile: 0, ±90, ±180, ±270 (în realitate acestea specifică numai patru orientări diferite).

15. `String` Şir de caractere

Această proprietate defineşte şirul de caractere care urmează să fie afişat de obiectul text.

16. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul text, şirul de caractere este întotdeauna *„text"*.

17. `Units` inches | centimeters | normalized | points | pixels | data

Această proprietate precizează unităţile de măsură utilizate pentru interpretarea dimensiunilor şi localizarea datelor. În toate cazurile măsurarea se face din colţul stânga-jos al ferestrei grafice. Unităţile normalizate transformă colţul din stânga-jos al ferestrei grafice în coordonatele (0,0), iar colţul din dreapta-sus în (1.0, 1.0). Celelalte unităţi de măsură sunt unităţi absolute, iar unitatea punct (points) este egală cu 1/72 inch.

18. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului; poate fi afişată folosind funcţia *get*.

19. `VerticalAlignment` top | cap | middle | baseline | bottom

Această proprietate specifică tipul de aliniere a textului pe verticală faţă de punctul menţionat de proprietatea `Position`. Implicit, textul este centrat (middle).3

20. `Visible` on | off

Această proprietate determină dacă obiectul este sau nu vizibil pe ecran. Dacă proprietatea `Visible` este stabilită pe „off", textul nu este afişat.

25.2. Crearea unui obiect grafic linie

Pentru crearea obiectelor linie se foloseşte funcţia grafică de nivel scăzut *line*; se apelează cu una dintre sintaxele:

h= *line*(x,y) h=*line*(x,y,z)

h=*line*('Nume_proprietate', Valoare_ proprietate, ...)

line(x,y) - adaugă liniile din vectorii x şi y în planul axelor curente. Dacă x şi y sunt matrice, se trasează câte o linie pentru fiecare coloană.

line(x,y,z) - creează linii în coordonate 3D.

Obiectele linie sunt copii ai obiectelor axe. Liniile sunt elemente fundamentale folosite pentru realizarea graficelor, contururilor şi marginilor suprafeţelor. Funcţia *line* acceptă ca argumente de intrare perechile nume / valoare_proprietate. Aceste proprietăţi, care controlează diferite aspecte ale obiectului linie, sunt descrise în continuare şi pot fi stabilite sau citite, utilizând funcţiile *set* şi *get*.

Perechile (x,y) sau (x,y,z) pot fi urmate de perechile nume / valoare_proprietate pentru a specifica proprietăţi suplimentare ale obiectului linie. Pentru modificarea unui obiect creat anterior coordonatele pot fi omise, specificându-se numai perechile nume/valoare_proprietate,.

Dacă se foloseşte cu argument de ieşire, funcţia returnează un identificator h, folosit pentru controlul caracteristicilor obiectului linie.

25.2.1. Proprietăţile obiectului grafic linie

Proprietăţile obiectului linie pot fi specificate la crearea acestuia, prin includerea perechilor nume/valoare_proprietate sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `Children` vector al identificatorilor

Obiectul linie nu are copii, atfel încât această proprietate este întotdeauna o matrice goală.

2. `Clipping` on | off

Dacă proprietatea `Clipping` este stabilită pe „on", orice porţiune de linie din afara zonei definită de dreptunghiul axelor nu va fi afişată. Valoarea implicită este „on".

3. `Color` Culoarea specificată

Această proprietate precizează culoarea folosită pentru trasarea liniei. Culoarea poate fi specificată utilizând un vector al valorilor RGB sau un nume de culoare predefinită în MATLAB.

4. `EraseMode` normal | background | xor | none

Această proprietate controlează metodele folosite de MATLAB pentru desenarea şi ştergerea liniilor; este utilizată pentru crearea secvenţelor de animaţie, în care, pentru a obţine efectele dorite, este necesar controlul individual al modului de redesenare a fiecărui obiect. Modul normal afectează întreaga zonă afişată şi permite obţinerea unei imagini foarte bune, dar necesită mult timp. Celelalte moduri sunt mai rapide, dar rezultă imagini calitativ mai slabe.

5. `LineStyle` - | -- | : | -. | + | o | * | . | x

Această proprietate determină tipul de linie sau marker care este folosit pentru trasarea obiectului linie. Sunt acceptate următoarele tipuri de linii şi markere (cu simbolurile precizate în paranteze): linie continuă (-), linie întreruptă (--), linie-punct (-.), linie din puncte (:), cerc (o), plus (+), punct (.), steluţă (*) şi x (x).

6. `LineWidth` scalar

Această proprietate precizează grosimea liniilor cu care se face reprezentarea. Valoarea implicită este 1 punct (1/72 inch).

7. `MarkerSize` scalar

Această proprietate precizează dimensiunea marcajelor; se aplică numai dacă proprietatea LineStyle este stabilită pe: cerc (o), plus (+), punct (.), steluţă (*) şi x (x). Dimensiunea implicită este 6 puncte (1 punct = 1/72 inch).

8. `Parent` identificator (numai de citit)

Această proprietate returnează identificatorul părintelui obiectului linie, care este întotdeauna un obiect axe.

9. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul linie, şirul de carcatere este întotdeauna „*line*".

10. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului; poate fi afişată folosind funcţia *get*.

11. `Visible` on | off

Această proprietate determină dacă obiectul este sau nu vizibil pe ecran. Dacă proprietatea `Visible` este setată `off`, obiectul linie nu este vizibil.

12. `Xdata` vector
13. `Ydata` vector
14. `Zdata` vector

Aceste proprietăţi specifică datele utilizate pentru a genera liniile. Dacă datele sunt sub forma unor matrice, pentru fiecare coloană se trasează câte o linie. În acest caz toate cele trei matrice trebuie să aibă acelaşi număr de linii.

25.3. Crearea unui obiect grafic suprafaţă

Pentru crearea obiectelor grafice suprafaţă se foloseşte funcţia *surface*; se apelează cu una dintre sintaxele :

h=*surface*(X,Y,Z,C) h=*surface*(Z,C)
h=*surface*(X,Y,Z) h=*surface*(C)
h=*surface*('Nume_proprietate', Valoare_proprietate,...)

Surface este o funcţie grafică de nivel scăzut pentru crearea obiectelor suprafaţă, Z(X,Y), care sunt copii ai obiectului axe. X şi Y sunt vectorii coordonate, iar indicii matricei de culoare sunt precizaţi de matricea C. Dacă matricea de culoare este omisă se consideră C=Z, adică culoarea este proporţională cu înălţimea. Limitele axelor x şi y sunt date de elementele din matricele X şi Y, iar dinamica scalei de culoare este precizată în matricea C.

Funcţia *surface* acceptă perechile nume/valoare_proprietate ca argumente de intrare. Aceste proprietăţi, prin care se pot controla diferite aspecte ale obiectului, sunt descrise în continuare. Acestea pot fi stabilite sau citite după ce obiectul a fost creat, utilizând funcţiile *set* şi *get*.

Dacă x este un vector de lungime n, y un vector de lungime m, iar C o matrice m×n, colţurile fiecărei zone a suprafeţei sunt precizate de tripletele (x(j),y(i),Z(i,j)), unde x este asociat coloanelor matricei Z, iar y este asociat liniilor acesteia.

25.3.1. Proprietăţile obiectului grafic suprafaţă

Proprietăţile obiectului suprafaţă pot fi specificate la crearea acestuia prin includerea perechilor nume/valoare_proprietate sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `Cdata` matrice

Această proprietate precizează culoarea pentru fiecare punct al matricei `ZData`. Dacă `CData` nu are aceeaşi dimensiune ca `ZData`, atunci suprafaţa este tratată ca o textură.

2. `Children` vector al identificatorului de control

Obiectele suprafaţă nu au copii, astfel că acest vector este gol.

3. `Clipping` on | off

Dacă proprietatea `Clipping` este stabilită pe „on", orice porţiune de suprafaţă din afara zonei definită de dreptunghiul axelor nu va fi afişată.

4. `EdgeColor` Culoare_specificată | non | flat | interp

Prin această proprietate se precizează culoarea marginilor (gridului) ariilor rectangulare definite de matricea Zdata, funcţie de opţiune: o singură culoare pentru toate marginile ariilor elementare (Culoare_specificată), transparente („non"), o aceeaşi culoare pentru fiecare margine de arie elementară („flat") sau o trecere graduală prin interpolarea liniară a culorilor din nodurile reţelei elementare a suprafeţei („interp").

5. `FaceColor` Culoare_specificată | non | flat | interp

Prin această proprietate se precizează culoarea suprafeţei şi modul de colorare a ariilor rectangulare definite de matricea Zdata, funcţie de opţiune: o singură culoare pentru toate ariile elementare (Culoare_specificată), transparente („non"), o singură culoare pentru fiecare arie elementară („flat") sau o trecere graduală prin interpolarea liniară a culorilor din nodurile reţelei elementare a suprafeţei („interp").

6. `LineStyle` tip de linie predefinită

Această proprietate determină tipul de linie pentru reprezentarea reţelei care defineşte suprafaţa (liniile de separare dintre ariile elementare). Sunt acceptate următoarele tipuri de linii (cu simbolurile precizate în paranteze): linie continuă (-); linie întreruptă (--) ; linie-punct (-.) ; linie din puncte (:).

7. MeshStyle both | row | column

Această proprietate precizează care dintre liniile reţelei de separare a suprafeţelor elementare se trasează: toate (both), numai cele corespunzătoare axei X (row) sau numai cele corespunzătoare axei Y (column). Sunt acceptate următoarele

tipuri de linii (cu simbolurile precizate în paranteze): linie continuă (-); linie întreruptă (--) ; linie-punct (-.) ; linie din puncte (:).

8. `Parent` identificator (numai de citit)

Această proprietate returnează identificatorul părintelui obiectului suprafaţă, care este un obiect axe.

9. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul suprafaţă şirul de caractere este întotdeauna *„surface"*.

10. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului, care poate să nu fie folosită, dar poate fi afişată folosind funcţia *get*.

11. `Visible` on | off

Această proprietate determină dacă obiectul este sau nu vizibil pe ecran. Dacă proprietatea `Visible` este stabilită pe „`off`", obiectul suprafaţă nu este vizibil (sau este şters), dar liniile axelor, marcajele de divizare şi etichetele nu sunt afectate.

12. `Xdata` vector | matrice

Proprietatea Xdata este un vector sau o matrice care specifică poziţia (coordonata) pe axa X a elementului corespunzător din matricea ZData.

13. `YData` vector | matrice

Proprietatea Ydata este un vector sau o matrice care precizează poziţia (coordonata) pe axa Y a elementului corespunzător din matricea ZData.

14. `ZData` matrice

Proprietatea Zdata este o matrice care precizează poziţia (coordonata) pe axa Z a punctelor suprafeţei.

25.4. Crearea unui obiect grafic imagine

Pentru crearea sau afişarea unui obiect imagine, se foloseşte funcţia *image*; se apelează cu una dintre sintaxele :

h=*image*(C) h=*image*(x,y,C)
h=*image*('Nume_proprietate',Valoare_proprietate,...)

image(C) - afişează matricea C ca imagine. Fiecare element al acestei matrice precizează culoarea unei zone dreptunghiulare a imaginii. Culoarea unei zone este determinată prin asocierea la fiecare element al matricei C a unui indice al matricei de culoare curente.

image(x,y,C) - afişează aceeaşi imagine ca image(C), cu limitele precizate de vectorii x şi y;

h=*image* - returnează un identificator de control al caracteristicilor obiectului creat.

Funcţia *image* este atât o instrucţiune de nivel înalt pentru afişarea imaginilor, cât şi una de nivel scăzut pentru crearea obiectelor imagine. Obiectul imagine este copil al obiectului axe. MATLAB afişează imaginea prin transformarea fiecărui element al

matricei cu funcţia *colormap*. Funcţia *image* crează obiecte grafice de tip imagine şi acceptă perechile nume/valoare_proprietate ca argumente de intrare. Aceste proprietăţi pot fi stabilite sau citite utilizând funcţiile *set* şi *get*.

Matricea argument poate fi urmată de perechile nume/valoare_proprietate, pentru a specifica proprietăţi suplimentare ale imaginii sau poate fi omisă, specificâdu-se numai proprietăţile care se doresc a fi modificate.

25.4.1. Proprietăţile obiectului grafic imagine

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate):

1. `CData`

Matricea cu datele de culoare a fiecărui element al imaginii. Funcţia *image*(C) asignează C la `Cdata`. Fiecare element al matricei `Cdata` specifică culoarea unei zone dreptunghiulare a imaginii. Elementele din `Cdata` sunt utilizate ca indici asociaţi unei matrice de culoare curentă. Valorile zecimale sunt rotunjite la cel mai mic întreg, iar valorile din afara intervalului [1 , *length*(*colormap*)] sunt omise, zona respectivă fiind transparentă.

2. `Clipping` on | off

Dacă proprietatea `Clipping` este stabilită pe „on", orice porţiune a imaginii care este în afara dreptunghiului descris de axe nu este afişată.

3. `Parent` identificator de control (numai pentru citire)

Proprietatea `Parent` transmite identificatorul de control al părintelui axelor imaginii. Axele sunt întotdeauna părinţi ai imaginii.

4. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul imagine, şirul de caratere este întotdeauna „*image*".

5. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului; poate fi văzută folosind funcţia *get*.

6. `Visible` on | off

Proprietatea `Visible` determină dacă obiectul este sau nu afişat pe ecran.

7. `Xdata` X[1 n]
8. `Ydata` Y[1 m]

Aceste proprietăţi precizează poziţia liniilor şi coloanelor reprezentate din matricea imaginii.

26. CONTROLUL CARACTERISTICILOR GRAFICE

Pentru controlul caracteristicilor grafice se folosesc următoarele funcţii MATLAB:

set	Stabileşte proprietăţile obiectelor;
get	Returnează proprietăţile obiectelor;
reset	Iniţializează proprietăţile obiectelor;
delete	Şterge obiectele grafice;
drawnow	Completează reprezentarea grafică şi o actualizează;
newplot	Stabileşte proprietatea NewPlot a obiectelor figură şi axe.

26.1. Stabilirea proprietăţilor obiectelor

Stabilirea proprietăţilor obiectelor se realizează cu funcţia *set*: se apelează cu una dintre sintaxele:

set(h)
v=*set*(h, 'Nume_proprietate')
v=*set*(h, 'Nume_proprietate', Valoare_proprietate, ...)

Funcţia *set* se poate utiliza pentru modificarea valorii proprietăţilor obiectului al cărui identificator este h (cu excepţia proprietăţilor care numai se citesc). Funcţia acceptă mai multe perechi nume/valoare_proprietate la o singură apelare. Pentru stabilirea valorilor implicite ale proprietăţilor obiectului se concatenează cuvântul `Default` cu tipul de obiect şi numele proprietăţii. Spre exemplu, pentru a stabili culoarea axelor din figura curentă la valoarea implicită se utilizează instrucţiunea:

set(*gcf*, '`DefaultAxesColor`', [1 1 1])

Apelată cu sintaxa *set*(h), funcţia returnează o listă cu toate proprietăţile obiectului cu identificatorul h, precum şi valorile acceptate de acestea. Valorile curente sunt cele dintre acolade. Dacă sintaxa folosită este v=*set*(h, '`Nume_proprietate`'), funcţia returnează o listă cu valorile posibile ale proprietăţii specificate.

Funcţia *set* acceptă şirurile de caractere `default`, `factory` şi `remove` ca valori de proprietate. Specificarea valorii `default` stabileşte ca valoare implicită prima valoare definită în lista proprietăţii respective. Spre exemplu, următoarea secvenţă stabileşte proprietatea `EdgeColor` la verde:

```
h=surf(peaks)
set(0,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')
```

Parametrul `factory` stabileşte proprietatea la caracteristica impusă. Spre exemplu, secvenţa:

```
h=surf(peaks)
set(0,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','factory')
```

stabileşte marginile suprafeţei cu identificatorul h la negru, valoare definită implicit pentru `factory`.

Exemplul 26.1.1. Să se stabilească limitele axelor la valorile x=[10, 40], y=[20, 50] şi z=[-5, 10], folosind proprietăţile corespunzătoare ale obiectului axe.
Soluţia este:

```
set(gca,'XLim',[-10 60],'YLim',[-5 40],...
    'ZLim',[-15 15])
```

Exemplul 26.1.2. Listaţi valorile acceptate de proprietatea `EdgeColor` a obiectului suprafaţă.
Cu secvenţa MATLAB:

```
set(h,'EdgeColor')
```

se obţine: [none | {flat} | interp] -or- a ColorSpec

Exemplul 26.1.3. Listaţi proprietăţile obiectului axe.
Cu secvenţa MATLAB:

```
h=surf(peaks)
set(h)
```

se obţine:

CData
EdgeColor: [none | {flat} | interp] -or- a ColorSpec.
EraseMode: [{normal} | background | xor | none]
FaceColor: [none | {flat} | interp | texturemap] -or- a ColorSpec.
LineStyle: [{-} | -- | : | -. | + | o | * | . | x]
LineWidth
MarkerSize
MeshStyle: [{both} | row | column]
XData
YData
ZData
ButtonDownFcn
Clipping: [{on} | off]
Interruptible: [{no} | yes]
Parent
UserData
Visible: [{on} | off]

26.2. Citirea proprietăţilor obiectelor

Citirea valorilor curente ale proprietăţilor unui obiect se realizează cu funcţia *get*, se apelează cu una dintre sintaxele:

get(h) p=*get*(h) v=*get*(h, 'Nume_proprietate')

unde h este identificatorul obiectului a cărui proprietate se citeşte.

- *get*(h) - returnează o listă cu toate proprietăţile obiectului având identificatorul h;
- v=*get*(h, 'Nume_proprietate') - returnează valoarea curentă a proprietăţii specificate a obiectului cu identificatorul h;

Pentru a obţine valoarea implicită a proprietăţii `Color` a obiectului figură se utilizează următoarea secvenţă:

```
get(0,'DefaultFigureColor')
```


Lista completă a tuturor valorilor curente pentru obiectele descendente (copii) ale obiectului cu identificatorul h, se obţine cu instrucţiunea:

```
get(h,'Default')
```

Exemplul 26.2.1. Fie reprezentarea grafică:

```
surf(peaks)
text(26,50,7,'Valoarea maxima')
```

Pentru citirea liniilor 1 şi 2 ale identificatorului h al obiectului se foloseşte secvenţa:

```
h=get(gca,'Children')
a=get(h(1),'Type')
b=get(h(2),'Type')
```

obţinându-se:

a=text
b=surface

Pentru stabilirea culorilor obiectului „text", cu identificatorul h(1), se foloseşte instrucţiunea:

```
set(h(1),'Color', [1 0 0])
```

iar pentru obţinerea listei proprietăţilor suprafeţei, precum şi a valorilor curente ale acesteia, se foloseşte instrucţiunea:

```
get(h(2))
```

obţinându-se rezultatul:

CData = [(49 by 49)]
EdgeColor = [0 1 0]
EraseMode = normal
FaceColor = flat
LineStyle = -
LineWidth = [0.5]
MarkerSize = [6]
MeshStyle = both
XData = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49]
YData = [(49 by 1)]
ZData = [(49 by 49)]
ButtonDownFcn =
Children = []
Clipping = on
Interruptible = no
Parent = [0.000366211]
Type = surface
UserData = []
Visible = on

26.3. Iniţializarea proprietăţilor obiectelor

Iniţializarea tuturor proprietăţilor axelor sau figurilor la valorile implicite, cu excepţia poziţiei, se realizează cu funcţia *reset;* se apelează cu sintaxa:

reset(h)

unde h este identificatorul de control al caracteristicilor proprietăţilor.
Spre exemplu:

reset(*gca*) - iniţializează proprietăţile axelor curente;
reset(*gcf*) - iniţializează proprietăţile figurii curente.

26.4. Actualizarea şi ştergerea obiectelor

Pentru actualizarea şi completarea unei reprezentări grafice, realizată în mai multe etape se foloseşte funcţia *drawnow* ; se apelează cu sintaxa:

drawnow

Următoarele acţiuni determină MATLAB să reactualizeze reprezentarea grafică:

- revenirea la prompterul MATLAB;
- folosirea comenzii *pause;*
- executarea unei comenzi *getframe;*
- executarea comenzii *drawnow.*

Pentru trecerea la o nouă figură atunci când se utilizează funcţiile de nivel scăzut pentru crearea obiectelor grafice, se foloseşte funcţia *newplot*, care este un antet al identificatorului proprietăţii `NextPlot` a fişierelor grafice M; se apelează cu sintaxa:

h=*newplot*

Funcţia *nextplot* realizează:

1. deschiderea unei figuri noi, dacă proprietatea `NextPlot` a obiectului figură este „`New`";
2. ştergerea şi iniţializarea figurii curente, dacă proprietatea `NextPlot` a obiectului figură este „`Replace`";
3. deschiderea unor axe noi, dacă proprietatea `NextPlot` a obiectului axe este „`New`";
4. şteargerea şi iniţializarea axelor curente, dacă proprietatea `NextPlot` a obiectului axe este „`Replace`".

Ştergerea ferestrelor grafice sau a obiectelor din acestea definite de identificatorul h se face cu funcţia *delete;* se apelează cu sintaxa:

delete(h)

Dacă identificatorul este al obiectului figură, fereastra grafică este închisă fără confirmare, iar obiectul este şters.

27. CONTROLUL CULORILOR REPREZENTĂRILOR GRAFICE

27.1. Controlul culorilor reprezentărilor grafice

Funcţiile MATLAB folosite pentru controlul culorilor reprezentărilor grafice sunt:

brighten Controlează luminozitatea matricei de culoare;
caxis Stabileşte limitele între care se scalează matricea de culoare pentru reprezentarea grafică;
colormap Stabileşte sau returnează matricea de culoare folosită pentru reprezentarea grafică;
contrast Controlează contrastul matricei de culoare;
hsv2rgb Converteşte nuanţele de culori saturate în culorile roşu-verde-albastru;
pcolor Reprezintă matricea de pseudoculoare a resurselor hardware ale calculatorului;
rgb2hsv Converteşte culorile roşu-verde-albastru în nuanţe de culori saturate;
rgbplot Reprezintă grafic ponderile culorilor fundamentale (roşu-verde-albastru) în matricea de culoare;
shading Stabileşte modul de colorare al suprafeţelor;
spinmap Roteşte culorile reprezentării grafice (efecte speciale).

27.1.1. Specificarea culorii în MATLAB

Specificarea culorii cu care se realizează o reprezentare grafică în MATLAB se poate face prin unul din procedeele:

- cu numele scurt al culorii;
- cu numele lung al culorii;
- prin tripletul RGB (Red-Green-Blue).

Numele scurt sau lung sunt şiruri de caractere care specifică una din cele opt culori predefinite. Tripletul RGB este un vector linie cu trei elemente, care specifică intensitatea celor trei componente fundamentale ale oricărei culori (roşu-verde-albastru). Elementele vectorului trebuie să fie numere în intervalul [0.0 1.0].

Cele trei forme de apelare pentru culorile predefinite în MATLAB sunt date în tabelul 27.1.

Următoarele trei secvenţe sunt echivalente, fiecare reprezentând graficul y(x) cu o linie de culoare verde:

- `plot(x,y,'g')`
- `plot(x,y,'green')`
- `h=plot(x,y); set(h,'Color',[0,1,0])`

Ultima variantă este o formă mai rafinată de folosire a caracterelor de control al caracteristicilor grafice, care se realizează prin setarea parametrului h cu care se operează ulterior.

Tabelul 27.1

Numele scurt	Numele lung	Tripletul RGB
y	yellow	1,1,0
m	magenta	1,0,1
c	cyan	0,1,1
r	red	1,0,0
g	green	0,1,0
b	blue	0,0,1
w	white	1,1,1
k	black	0,0,0

Exemplul 27.1.1. Să se schimbe culoarea fondului unei reprezentări grafice (background) în roz.
Soluţia este secvenţa MATLAB:

```
set(gcf ,'Color',[1, .4, .6])
```

27.1.2. Stabilirea matricei de culoare

Matricea de culoare pentru o reprezentare grafică este o matrice cu dimensiunea m x 3, ale cărei elemente sunt numere reale în intervalul [0.0 1.0]. Linia k a unei matrice de culoare defineşte culoarea asociată indicelui k dintr-o scală a culorilor; ea este specificată prin intensitatea culorilor fundamentale (tripletul RGB): C(k,:)=[r(k) g(k) b(k)].

Funcţia *colormap* prezintă matricea de culoare folosită de reprezentarea grafică; se apelează cu una dintre sintaxele:

colormap(C) *colormap*('default') C=*colormap*

- *colormap*(C) - stabileşte matricea de culoare C;
- C=*colormap* - returnează matricea de culoare curentă;
- *colormap*('default') şi *colormap*(*hsv*) - setează matricea de culoare la valoarea implicită, modelul HSV (nuanţe de culoare saturate). În MATLAB mai există următoarele modele predefinite de matrice de culoare: `gray,hot,cool,cooper,pink`.

Dacă dimensiunea matricei de culoare nu este specificată ca argument de intrare, atunci aceasta este considerată egală cu 64. Efectul acesteia este obţinerea unui număr de nuanţe de culoare egal cu numărul introdus ca argument.

Exemplul 27.1.2. Pentru reprezentarea tonurilor de culoare ale unei matrice cu n=16 nuanţe de culoare, se foloseşte secvenţa MATLAB:

```
n=16;N=[1:n;1:n];
subplot(9,1,1); colormap(hsv);surface(N)
```

27.1.3. Scalarea matricei de culoare

Funcţia *caxis* permite modificarea proprietăţilor `CLim` şi `CLimMode` ale obiectului axe creat cu funcţia *axes*. Transformările de culoare sunt controlate prin funcţia *colormap* şi proprietăţile `CLim` şi `CLimMode`.

Funcţia *caxis* scalează axa pseudocolorii matricei de culoare; se apelează cu una dintre sintaxele:

`caxis`([cmin cmax]) - stabileşte parametrii `CLim` de scalare ai axei de culoare la valorile precizate. Culorile din afara intervalului cmin≤ c≤cmax sunt omise; astfel, zonele sau numai pixelii corespunzători acelor culori, nu sunt reprezentate;

`caxis`('auto') - stabileşte implicit CLimMode, scalând automat parametrii între valoarea minimă şi maximă a axei de culoare. Prin această comandă numai valorile `Inf` şi `NaN` sunt omise (nereprezentate);

v=*caxis* - returnează un vector linie cu două elemente, care conţine limitele [cmin cmax] ale matricei de culoare curente;

caxis(caxis) - „îngheaţă" scalarea axei de culori la limitele curente, astfel încât următoarele reprezentări se pot realiza folosind aceeaşi dinamică de culoare, pentru a compara eventualele evoluţii codificate după culoare.

Exemplul 27.1.3. Să se reprezinte o sferă de rază r=1:
a) folosind întreaga scală de culoare;
b) vizualizând numai jumătatea inferioară;
c) folosind numai prima jumătate a scalei de culoare.

Soluţia corespunde secvenţei MATLAB:

```
[X, Y,Z]=sphere(32);
C=Z; %matricea de culoare
subplot(221);
surf(X,Y,Z,C); % pentru punctul a
```

În acest caz valoarea minimă a matricei de culoare este -1, iar valoarea maximă este +1. Reprezentarea grafică conţine întreaga sferă şi foloseşte toate culorile din matricea de culoare a modelului ales.

Pentru a vizualiza numai o zonă dintr-o reprezentare grafică 3D, folosind însă toate culorile din matricea de culoare a modelului ales, trebuie ca valorile asociate matricei Z să fie între limitele minime şi maxime ale matricei de culoare C (precizate cu funcţia *caxis*), ceea ce este echivalent cu zonele de interes. Pe acest principiu se pot construi **filtre de culoare**. Deoarece în acest exemplu se doreşte reprezentarea numai a semisferei inferioare, pentru care matricea Z are elementele în intervalul [-1, 0], se foloseşte secvenţa:

```
subplot(222); surf(X,Y,Z,C);
caxis([-1 0]); % pentru punctul b
```

Pentru a comprima scala de culoare a reprezentării, folosindu-se numai prima jumătate din dinamica acesteia, dar reprezentându-se întreaga sferă, se modifică limitele cmin şi cmax ale funcţiei *caxis*, astfel încât valorile matricei de culoare C să se situeze în poziţia relativă dorită. Deoarece valorile extreme ale matricei C sunt [-1, 1], noile limite impuse prin caxis sunt [-1, 3]. Pentru cazul c din acest exemplu, se foloseşte secvenţa:

```
subplot(223); surf(X,Y,Z,C);
caxis([-1 3]); % pentru punctul c
```

Instrucţiunea `caxis('auto')` iniţializează limitele scalei de culori la dinamica maximă, cu instrucţiunea z=*caxis* obţinându-se vectorul v=[-1 1].

27.1.4. Stabilirea modului de colorare

Funcţia *shading* setează modurile de colorare a obiectelor grafice suprafaţă (*surface*) şi zonă (*patch*); se apelează cu una dintre sintaxele:

shading `faceted` - stabileşte modul de reprezentare a zonelor şi suprafeţelor cu aceeaşi culoare, liniile reţelei (mesh-ului) fiind vizibile;

shading `flat` - stabileşte reprezentarea grafică a zonelor cu culoarea specificată pentru vârful din stânga-sus al reţelei, liniile reţelei (mesh-ului) fiind invizibile;

shading `interp` - stabileşte culoarea de reprezentare a fiecărui segment sau arie elementară la o nuanţă rezultată din interpolarea liniară a culorilor capetelor liniei sau colţurilor suprafeţei.

Funcţia *shading* stabileşte proprietăţile EdgeColor şi FaceColor pentru toate obiectele suprafaţă şi zonă din axele curente, adică pentru cele create cu funcţiile *mesh*, *surf*, *pcolor*, *fill* şi *fill3*.

27.1.5. Matricea de pseudoculoare

Funcţia *pcolor* reprezintă pseudoculoarea unei matrice de culoare; se apelează cu una dintre sintaxele:

pcolor(C) - returnează o reprezentare grafică cu o suprafaţă de celule colorate în culorile precizate de matricea C. Modul implicit este '*faceted*', culoarea unei celule fiind constantă, caz în care ultima linie şi coloană a matricei C nu sunt utilizate. Trecerea de la matricea C la matricea de culoare este realizată de funcţiile *colormap* şi *caxis*;

pcolor(x,y,C) - unde x este un vector cu n elemente, y un vector cu m elemente, iar C o matrice m x n (x este asociat coloanelor, iar y este asociat liniilor) - returnează o reprezentare grafică cu o suprafaţă divizată în m linii şi n coloane, nu obligatoriu echidistante, cu celulele colorate în culorile precizate de matricea C;

pcolor(X,Y,C) - unde X, Y şi C sunt matrice de aceleaşi dimensiuni - returnează o reprezentare grafică cu o suprafaţă divizată în celulele. Celula (i,j), localizată de [X(i,j), Y(i,j)] este colorată cu C(i,j), dacă funcţia *shading* (haşura) este în modul '*faceted*' sau '*flat*' (aceeaşi culoare), sau cu o pseudoculoare, rezultată prin interpolarea culorilor din cele patru colţuri ale celulei, dacă setarea haşurii este '*interp*';

h=*pcolor*(...) - returnează un caracter de control al caracteristicilor obiectului suprafaţă (creat cu funcţia *surface*), care este copil al obiectului axe (creat cu funcţia *axes*).

Funcţia *pcolor*(X,Y,C) este echivalentă reprezentării unei suprafeţe *surf*(X,Y,0*Z,C) văzută de deasupra, adică din poziţia *view*([0 90]). Funcţiile *pcolor* şi *image* sunt similare, însă *pcolor*(C) specifică culorile colţurilor, în timp ce *image*(C) specifică culorile celulelor, indexate direct din matricea de culoare nescalată. Prin urmare, numărul de colţuri pentru *pcolor*(C) este acelaşi cu numărul celulelor pentru

image(C). Cu funcţia *pcolor*(X,Y,C) se poate obţine o reprezentare cu reţea de linii (grid), lucru care nu este posibil cu funcţia *image*.

Exemplul 27.1.4. Urmăriţi modul de utilizare şi efectul următoarelor două secvenţe:

```
pcolor(hadamard(20))
colormap(gray(2))
n=6; r=(0:n)'/n; t=pi*(-n:n)/n;
X=r*cos(t); Y=r*sin(t); C=r*cos(2*t);
pcolor(X,Y,C); axis('square')
```

27.1.6. Conversia culorilor RGB-HSV şi HSV-RGB

Pentru a converti o matrice de culoare codificată în culorile fundamentale (RGB) într-o matrice codificată nuanţă-saturaţie-luminozitate (HSV) se foloseşte funcţia *rgb2hsv*; se apelează cu sintaxa:

H=*rgb2hsv*(M)

Exemplul 27.1.5. Să se convertească matricea de culoare:

$$M = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ .5 & .5 & .3 \\ 0 & .5 & .1 \end{bmatrix}$$

exprimată prin triplete RGB, într-o matrice de culoare exprimată prin tripletele HSV.
Cu secvenţa MATLAB:

```
M=[1 1 0;0 1 1;.5 .5 .3;0 .5 .1];
H=rgb2hsv(M)
```

se obţine rezultatul:

```
H =
   0.1667   1.0000   1.0000
   0.5000   1.0000   1.0000
   0.1667   0.4000   0.5000
   0.3667   1.0000   0.5000
```

Pentru conversia unei matrice de culoare codificată nuanţă-saturaţie-luminozitate (HSV) într-o matrice de culoare reprezentată prin tripletul RGB se foloseşte funcţia *hsv2rgb*; se apelează cu sintaxa:

M= *hsv2rgb*(H)

1. Dacă prima coloană a lui H (nuanţa culorii), H(:,1), variază de la 0 la 1, scala culorilor rezultate este **roşu-galben-verde-bleumarin-albastru-mov-roşu**;
2. Dacă a doua coloană a lui H (saturaţia), H(:,2), este zero, culorile sunt nesaturate, fiind cu nuanţe de gri, iar dacă este 1, culorile sunt saturate, adică nu conţin componenta de culoare albă;
3. Dacă a treia coloană a lui H (valoarea culorii), H(:,3), variază de la 0 la 1, luminozitatea creşte.

Exemplul 27.1.6. Să se convertească matricea de culoare:

$$H=\begin{bmatrix} 0.1667 & 1 & 0 \\ 0.5000 & 1 & 1 \\ 0.1667 & .4 & .5 \\ 0.3667 & 1 & .5 \end{bmatrix}$$

exprimată prin triplete HSV, într-o matrice de culoare exprimată prin tripletele RGB.
Cu secvenţa MATLAB:

```
H=[.1667 1 1;.5 1 1;.1667 .4 .5;.3667 1 .5];
M=hsv2rgb(H)
```

se obţine rezultatul:

```
M =
   0.9998   1.0000   0
   0        1.0000   1.0000
   0.5000   0.5000   0.3000
   0        0.5000   0.1001
```

Acest exemplu arată reciprocitatea funcţiilor de conversie a matricelor de culoare *rgb2hsv* şi *hsv2rgb*.

27.1.7. Controlul contrastului şi luminozităţii

Pentru a schimba contrastul unei imagini se foloseşte funcţia *contrast;* se apelează cu sintaxa:

C=*contrast*(X,M)

unde X este matricea imaginii asupra căreia se operează, iar M este numărul de nivele de gri. Acţiunea efectivă se face cu secvenţa:

image(X); *colormap*(C)

Pentru a controla luminozitatea (strălucirea) matricei de culoare se foloseşte funcţia *brighten;* se apelează cu una dintre sintaxele:

brighten(L) - înlocuieşte matricea de culoare curentă cu una mai luminoasă, dacă 0<L<1, sau mai întunecată, dacă -1<L<0, dar care are aceleaşi culori de bază; *brighten*(L) urmat de *brighten*(-L) reface culoarea originală;

C=*brighten*(L) - returnează o versiune a matricei de culoare curente, mai luminoasă sau mai întunecată, fără a opera asupra culorii reprezentării grafice;

Cn=*brighten*(C,L) - returnează o versiune a matricei de culoare C, mai luminoasă sau mai întunecată, fără a opera asupra culorii reprezentării grafice.

27.1.8. Rotirea culorii reprezentării grafice

Funcţia *spinmap* realizează rotirea ciclică a culorilor reprezentării grafice; se apelează cu una dintre sintaxele :

spinmap - realizează rotirea ciclică a culorilor matricei de culoare a reprezentării grafice la circa 5 secunde;

spinmap(t) - realizează rotirea ciclică a culorilor matricei de culoare timp de t secunde;

spinmap(t,dt) - realizează rotirea ciclică a culorilor matricei de culoare timp de t secunde, cu pasul dt. Spre exemplu dt=1 realizează o rotire cu viteză mai mică decât dt=2, iar dt=-2 realizează o rotire în sens invers;

spinmap(`inf`) - realizează o rotire a culorilor în timp infinit, buclă din care se iese cu „CTRL-C".

27.1.9. Reprezentarea grafică a ponderii culorilor fundamentale

Funcţia *rgbplot* returnează o reprezentare grafică a ponderii celor trei componente de culoare, roşu-verde-albastru, care constituie matricea de culoare C; se apelează cu sintaxa:

rgbplot(C)

Matricea C este o matrice m x 3 care este folosită de funcţia *colormap*. Fiecare coloană a matricei de culoare este reprezentată cu culoarea proprie.

Exemplul 27.1.7. Să se reprezinte graficul ponderilor componentelor RGB în matricea de culoare a modelului copper.
Cu secvenţa MATLAB:

```
rgbplot(copper); grid
```

se obţine graficul din figura 27.1.

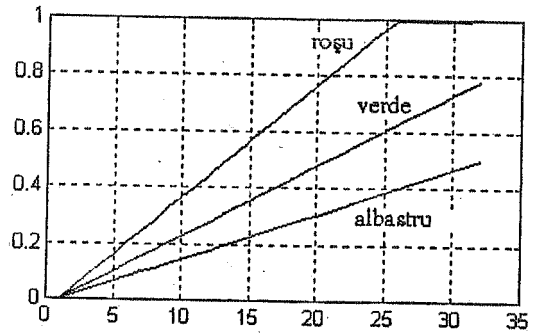


Fig. 27.1. Graficul ponderii culorilor fundamentale (RGB), în matricea de culoare a modelului *copper*.

27.2. Controlul iluminării reprezentărilor grafice

Funcţiile MATLAB folosite pentru controlul iluminării reprezentărilor grafice sunt:

surfnorm Reprezintă grafic normalele la suprafaţă;
surfl Controlează nuanţele de culoare ale suprafeţelor 3D prin sursele de lumină;
specular Suprafeţe reflectante;
diffuse Suprafeţe cu reflexii difuze.

27.2.1. Reprezentarea grafică a normalei la suprafaţă

Funcţia *surfnorm* calculează şi afişează normala la suprafeţele 3D; se apelează cu una dintre sintaxele:

surfnorm(Z) şi

surfnorm(X,Y,Z) - reprezintă grafic suprafaţa descrisă de matricea Z (sau de matricele X, Y şi Z) şi normalele la aceasta. Pentru inversarea direcţiilor normalelor se apelează funcţia *surfnorm* cu matricele X, Y şi Z, transpuse, *surfnorm*(X', Y', Z').

[Nx, Ny, Nz]=*surfnorm*(...) - returnează matricea componentelor normale la o suprafaţă 3D, descrisă de matricea Z (sau de matricele X, Y şi Z).

Algoritmul de calcul al normalelor la suprafeţele reprezentate în 3D se bazează pe aproximarea bicubică a datelor din matricele X, Y şi Z.

Exemplul 27.2.1. Să se reprezinte vectorii normali la suprafaţa unui trunchi de con cu razele R=10 şi r=5.

Cu secvenţa MATLAB:

```
[x, y, z]=cylinder(5:10);
surfnorm(x,y,z)
```

se obţine figura 27.2.

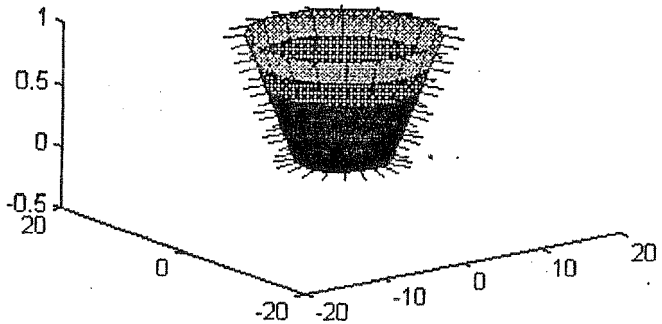


Fig. 27.2. Reprezentarea normalelor la suprafaţă cu funcţia *surfnorm*.

27.2.2. Controlul nuanţelor de culoare prin surse de lumină

Funcţia *surfl* reprezintă grafic suprafeţe 3D cu nuanţe de culoare bazate pe combinaţii ale modelelor de iluminare difuz (*difuse*), stălucitor (*specular*) şi normal. Suprafeţele pot fi vizualizate într-o culoare unică sau în culori interpolate; se apelează cu una dintre sintaxele:

surfl(Z) *surfl*(Z,s)

surfl(x, y, Z) *surfl*(x, y, Z, s)

Argumentul opţional s defineşte remanenţa, precizată printr-o matrice cu trei vectori s=[Sx, Sy, Sz], care specifică direcţia sursei de lumină. Aceasta poate fi dată şi

în coordonate sferice, s=[az elv], prin unghiul az în plan orizontal şi unghiul elv cu planul orizontal.

Exemplul 27.2.2. Să se reprezinte grafic funcţia *peaks* cu sursa de lumină plasată în poziţia [1, -4, 0.1].

```
surfl(x,y,z,[1, -4, .1]);
shading interp
colormap(gray)
axis([-3 3 -3 3 -8 8])
```

27.2.3. Reprezentarea suprafeţelor cu reflexii

Prin reflexie se înţelege partea de lumină reflectată de suprafaţă către observator. Valoarea acesteia variază de la 0.0 (fără reflexie) până la 1.0 (reflexie totală). Componentele vectorilor normali pot fi matrice pentru care normala este:

n(i,j) = [Nx(i,j), Ny(i,j), Nz(i,j)]

Aceste componente normale sunt calculate utilizând funcţia *surfnorm*. Pentru calculul coeficienţilor de reflexie ai suprafeţelor trebuie precizată poziţia sursei de lumină S şi poziţia V a observatorului faţă de obiectul a cărui suprafaţă reflectă. Cele două poziţii pot fi date în coordonate carteziene S=[Sx, Sy, Sz] şi V=[Vx, Vy, Vz] sau în coordonate cilindrice S=[azs, elvs] şi V=[azv, elvv].

Pentru calculul coeficientului de reflexie al suprafeţelor lucioase se utilizează funcţia *specular*, care foloseşte vectorii normali la suprafaţă; se apelează cu sintaxa:

r=*specular*(Nx, Ny, Nz, S, V)

iar pentru calculul coeficientului de reflexie al suprafeţelor difuze se utilizează funcţia *diffuse*, bazată pe legea lui Lambert; se apelează cu sintaxa:

r=*diffuse*(Nx, Ny, Nz, S)

De regulă, funcţia *specular* se utilizează ca parte a funcţiei *surfl* pentru a realiza suprafeţe cu efecte speciale (umbre şi reflexii). Algoritmul implementează o aproximare care conduce la efecte de suprafeţe metalice. Cea mai mare strălucire se produce dacă vectorul normal are direcţia (s+v)/2, unde s este direcţia sursei şi v direcţia observatorului.

27.3. Modele de culoare folosite în reprezentările grafice

bone	Reprezentarea în scala gri cu nuanţe de albastru;
cool	Reprezentarea cu nuanţe ale culorilor bleu şi mov;
copper	Reprezentarea cu nuanţe de arămiu;
flag	Reprezentarea în culorile roşu-alb-albastru-negru;
gray	Scala gri liniară pentru culoarea reprezentării;
hot	Reprezentarea în culorile negru-roşu-galben-alb;
hsv	Valoarea saturată a nuanţei culorii reprezentării;
jet	Variantă a HSV;
pink	Reprezentarea cu nuanţe de roz;
prism	Reprezentarea în culorile prismei.

27.3.1. Modelul „bone"

Funcţia *bone* returnează o matrice de culoare cu nuanţe de albastru; se apelează cu sintaxa:

C=*bone*(M) *colormap*(*bone*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.1. Reprezentarea culorilor modelului „bone" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(bone(n));surface(N)
```

27.3.2. Modelul „cool"

Funcţia *cool* returnează o matrice de culoare care conţine albastru-mov; se apelează cu sintaxa:

C=*cool*(M) *colormap*(*cool*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.2. Reprezentarea culorilor modelului „cool" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(cool(n));surface(N)
```

27.3.3. Modelul „copper"

Funcţia *copper* returnează o matrice de culoare cu nuanţe de arămiu; se apelează cu sintaxa:

C=*copper*(M) *colormap*(*copper*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.3. Reprezentarea culorilor modelului „cooper" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(copper(n));surface(N)
```

27.3.4. Modelul „flag"

Funcţia *flag* returnează o matrice de culoare care alternează roşu-alb-albastru-negru (cu perioada patru); se apelează cu sintaxa:

C=*flag*(M) *colormap*(*flag*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.4. Reprezentarea culorilor modelului „flag" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(flag(n));surface(N)
```

27.3.5. Modelul „gray"

Funcţia *gray* returnează o matrice de culoare într-o scală liniară de gri; se apelează cu sintaxa:

C=*gray*(M) *colormap*(*gray*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.5. Reprezentarea culorilor modelului „gray" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(gray(n));surface(N)
```

27.3.6. Modelul „hot"

Funcţia *hot* returnează o matrice de culoare care conţine negru-roşu-galben-alb; se apelează cu sintaxa:

C=*hot*(M) *colormap*(*hot*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.6. Reprezentarea culorilor modelului „hot" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(hot(n));surface(N)
```

27.3.7. Modelul „hsv"

Funcţia *hsv* returnează o matrice de culoare exprimată sub forma HSV (nuanţă-saturaţie-luminozitate): se apelează cu sintaxa:

C=*hsv* C=*hsv*(M) *colormap*(*hsv*)

Funcţia *hsv* se utilizează asociată cu funcţia *colormap* pentru a specifica numele matricei de culoare ale cărei componente variază. Scala de culori a acestui model de matrice de culoare este **roşu-galben-verde-bleumarin-albastru-mov-roşu**.

Funcţia *hsv*(m) returnează o matrice cu dimensiunea m x 3, care conţine matricea de culoare hsv. Fără argumentul m, funcţia *hsv* returnează o matrice cu dimensiunea 64x3.

Exemplul 27.3.7. Reprezentarea culorilor modelului „hsv" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(hsv(n));surface(N)
```

27.3.8. Modelul „jet"

Funcţia *jet* returnează o matrice de culoare în nuanţe saturate (variantă *hsv*); se apelează cu sintaxa:

C=*jet*(M) *colormap*(*jet*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.8.4. Reprezentarea culorilor modelului „jet" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(jet(n));surface(N)
```

27.3.9. Modelul „pink"

Funcţia *pink* returnează o matrice de culoare în nuanţele pastel; se apelează cu sintaxa:

C=*pink*(M) *colormap*(*pink*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.9. Reprezentarea culorilor modelului „pink" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(pink(n));surface(N)
```

27.3.10. Modelul „prism"

Funcţia *prism* returnează o matrice de culoare în nuanţele curcubeului (**roşu-oranj-galben-verde-albastru-violet**), care se repetă ciclic (cu perioada şase); se apelează cu sintaxa:

C=*prism*(M) *colormap*(*prism*)

unde M este un scalar, iar C este o matrice M x 3, ale cărei coloane reprezintă ponderile culorilor fundamentale, RGB.

Exemplul 27.3.10. Reprezentarea culorilor modelului „prism" cu n=16 trepte de culoare presupune secvenţa MATLAB:

```
clc;n=16;N=[1:n+1;1:n+1];subplot(9,1,1)
colormap(prism(n));surface(N)
```


28. MIŞCAREA ŞI ANIMAŢIA IMAGINILOR

Se fac referiri la următoarele funcţii MATLAB:

moviein Iniţializează memoria pentru salvarea cadrelor în mişcare;
getframe Returnează un cadru în mişcare;
movie Lansează operaţia de vizualizare a cadrelor unei imagini în mişcare.

Pentru a realiza mişcarea şi animaţia imaginilor (cadrelor) reprezentate într-o matrice M, se procedează în prima etapă la crearea unei noi matrice cu coloanele deplasate, utilizând funcţia *getframe*; se apelează cu una dintre sintaxele:

M=*getframe* - returnează un vector coloană cu un cadru mişcat, care reprezintă o fotografiere instantanee (pixel cu pixel). De obicei, *getframe* se utilizează într-o buclă *for* pentru a asambla o imagine M (matrice) în mişcare, care se foloseşte apoi cu funcţia *movie*;

M=*getframe*(h) - returnează un identificator care permite modificarea proprietăţilor obiectului figură sau a obiectului axe;

M=*getframe*(h,poz) - returnează un cadru mişcat al obiectului cu identificatorul h din zona încadrată de dreptunghiul cu dimensiunile (L,H), situat la distanţa (l,h) de colţul din stânga-jos al ferestrei grafice (în unităţile de măsură curente). Parametrul „poz" trebuie să fie un vector linie cu patru elemente:

poz=[l h L H]

Pentru a evita utilizarea unei memorii considerabile, înaintea realizării acestei operaţii este preferabilă prealocarea unei matrice de mişcare, capabilă să memoreze cele n cadre succesive. Pentru aceasta se generează anticipat o matrice de zerouri cu funcţia *moviein*; se apelează cu sintaxa:

M=*moviein*(n)

În realitate, funcţia *moviein* este un fişier-M cu o singură linie:

M=*zeros*(*length*(*getframe*),n)

După aceste prelucrări asupra matricei ce reprezintă imaginea de animat, se utilizează funcţia *movie* pentru a mişca cadrele într-o anumită ordine, cu o anumită frecvenţă şi un interval de timp precizat; se apelează cu una dintre sintaxele:

movie(M) - realizează mişcarea cadrelor matricei M, o singură dată. Matricea M trebuie să fie o matrice ale cărei coloane sunt cadrele mişcate ale unei imagini (realizată de obicei cu funcţia *getframe*);

movie(M, n) - realizează mişcarea matricei M de n ori. Dacă n este negativ, mişcarea se face o singură dată înainte şi o singură dată înapoi. Dacă n este un vector, primul element specifică de câte ori se face mişcarea, iar elementele 2, 3, ... specifică ordinea de prezentare a cadrelor în mişcare din matricea M;

movie(M, n, f) - parametrul f precizează frecvenţa de mişcare, în cadre/secundă. Valoarea implicită este 12;

movie(h,...) - realizează o mişcare a obiectului h. Identificatorul h poate fi al unui obiect figură sau axe;

movie(h, M, n, f, loc) - realizează mişcarea de n ori cu frecvenţa f cadre/secundă a obiectului cu identificatorul h, cu imaginea de bază în matricea M, localizat în poziţia loc=[x y] faţă de colţul din stânga-jos al ferestrei grafice.

Observaţie: Animaţia se poate realiza numai dacă memoria RAM video este de cel puţin 2Mo.

Exemplul 28.1. Să se scrie un program care să realizeze vibraţia funcţiei *peaks*.
Soluţia presupune secvenţa MATLAB:

```
z=peaks
surf(z)
lim=axis
M=moviein(20)
for j=1:20
      surf(sin(2*pi*j/20)*z, z)
      axis(lim)
      M(:,j)==getframe
end
movie(M,20)
```

Exemplul 28.2. Să se scrie un program care să realizeze vibraţia funcţiei predefinite *vibes* (membrană în forma de L).
Soluţia presupune secvenţa MATLAB:

```
load vibesdat
n = max(size(L1)); nh = fix(n/2);
x = (-nh:nh)/nh;
clear c; clf; view(3)
for k = 1:12
     eval(['c(k)=L' num2str(k) '(24,13)/3;'])
end
colormap(prism(6));
delt = 0.1; ncadre = 10;
M = moviein(ncadre);
for k = 1:ncadre
    disp(k)
    t=k*delt; s=c.*sin(sqrt(lambda)*t);
    L=s(1)*L1+s(2)*L2+s(3)*L3+s(4)*L4+s(5)*L5+s(6)*L6+...
    s(7)*L7+s(8)*L8+s(9)*L9+s(10)*L10+s(11)*L11+s(12)*L12;
    s=s .* lambda;
    V=s(1)*L1+s(2)*L2+s(3)*L3+s(4)*L4+s(5)*L5+s(6)*L6+...
    s(7)*L7+s(8)*L8+s(9)*L9+s(10)*L10+s(11)*L11+s(12)*L12;
    V(1:nh,1:nh) = NaN*ones(nh,nh);
    cla; surfc(x,x,L,V);
    axis([-1 1 -1 1 -1 1]); grid
    M(:,k) = getframe;
end
movie(M,20,10);
```


29. ÎNREGISTRAREA REPREZENTĂRILOR GRAFICE

Pentru înregistrarea imaginilor grafice se face apel la următoarele funcţii MATLAB:

orient — Returnează orientarea paginii pe care se face imprimarea („`portrait`", „`landscape`" sau „`tall`");
print — Tipăreşte graficele direct la imprimantă sau le salvează într-un fişier;
printopt — Returnează configuraţia locală a imprimatei.

Tipărirea reprezentărilor grafice la imprimantă se face cu funcţia *print* ; se apelează cu sintaxa:

print [-tip_driver] [-opţiune] [nume_fişier]

Apelată fără argumente, funcţia *print* trimite conţinutul ferestrei grafice curente la imprimanta implicită;

print nume_fişier - salvează reprezentarea din fereastra grafică curentă în fişierul „nume_fişier", în formatul corespunzător imprimantei implicite. Dacă există deja un fişier cu numele menţionat, prin această operaţie se înlocuieşte cel vechi. Cu opţiunea `-append`, noua reprezentare grafică este adăugată la vechiul conţinut. În cazul în care nu este precizată extensia fişierului care se salvează, acesta va lua extensia .ps sau .eps, funcţie de tipul de imprimantă specificat prin parametrul opţional -tip_driver.

- Tipurile de drivere acceptate direct sub Windows sunt:
  - dwin - trimite conţinutul ferestrei grafice la imprimanta curentă monocrom;
  - dwinc - trimite conţinutul ferestrei grafice la imprimanta curentă color;
  - dmeta - trimite conţinutul ferestrei grafice în clipboard, în format Metafile;
  - dbitmap - trimite conţinutul ferestrei grafice în **clipboard**, în format bitmap;
  - dsetup - returnează „caseta de dialog" Print Setup, care permite stabilirea parametrilor imprimantei, fără a tipări fereastra grafică.
- Driverele Postscript disponibile sunt:
  - dps - PostScript pentru imprimante monocrom;
  - dpsc - PostScript pentru imprimante color;
  - dps2 - Level 2 PostScript pentru imprimante monocrom;
  - dpsc2 - Level 2 PostScript pentru imprimante color;
  - deps - Encapsulated PostScript (EPSF);
  - depsc - Encapsulated Color PostScript (EPSF);
  - deps2 - Encapsulated Level 2 PostScript (EPSF);
  - depsc2 - Encapsulated Level 2 Color PostScript (EPSF).
- Driverele suplimentare, GhostScript sunt:
  - dlaserjet - HP LaserJet;
  - dljetplus - HP LaserJet+;
  - dljet2p - HP LaserJet IIP;

- dljet3 - HP LaserJet III;
- dcdeskjet - HP DeskJet 500C cu 1 bit/pixel color;
- dcdjcolor - HP DeskJet 500C cu 24 biţi/pixel color;
- dcdjmono - HP DeskJet 500C mono;
- ddeskjet - HP DeskJet şi DeskJet Plus;
- dpaintjet - HP PaintJet printer color;
- dpjetxl - HP PaintJet XL printer color;
- dbj10e - Canon BubbleJet BJ10e;
- dln03 - DEC LN03 printer;
- depson - Epson-compatibil (9 sau 24-pini);
- deps9high - Epson-compatibil 9-pini (rezoluţie mărită);
- depsonc - Epson LQ-2550 şi Fujitsu 3400/2400/1200 color;
- dgif8 - fişier format GIF 8-biţi color;
- dpcx16 - fişier format PCX (tip vechi), EGA/VGA - 16 culori;
- dpcx256 - fişier format PCX (tip nou), 256 culori.

MATLAB schimbă la tipărire culoarea neagră a fondului figurii în alb şi culoarea albă a axelor şi etichetelor în negru. Acestea pot fi menţinute la culorile folosite de reprezentarea grafică de pe ecran dacă se stabileşte pe „`off`" proprietatea `InvertHardCopy` a figurii, ca în exemplul:

```
set(gcf,'InvertHardCopy', 'off')
```

Pentru stabilirea orientării paginii pe care se face imprimarea conţinutului unei ferestre grafice se utilizează funcţia *orient*, se apelează cu una dintre sintaxele:

orient - returnează modul de imprimare pe pagină;

orient `landscape` - stabileşte modul de tipărire pe pagină, cu latura mică orientată pe verticală. Reprezentarea grafică ocupă întreaga pagină;

orient `portrait` - stabileşte modul de tipărire pe pagină, cu latura mare orientată pe verticală. Graficul ocupă un dreptunghi cu laturile în raportul 4/3, plasat la jumătatea paginii;

orient `tall` - stabileşte orientarea paginii „portrait", iar reprezentarea grafică ocupă întreaga pagină.

Înregistrările reprezentărilor grafice care să poată fi introduse ulterior în documente se pot transmite în **Cliboard**, cu una dintre opţiunile:

print `-dbitmap` *print* `-dmeta`

de unde sunt preluate şi prelucrate cu **Paintbrush**-ul sau cu alt utilitar destinat acestui scop.

Preluarea reprezentării grafice din Cliboard, cu Paintbrush-ul, se face în felul următor:

- se dă comanda *print* `-dbitmap` sau *print* `-dmeta` din MATLAB, cu reprezentarea care se doreşte preluată în fereastra grafică curentă;
- se apelează **Paintbrush**-ul, fără părăsirea MATLAB-ului;
- se foloseşte opţiunea PASTE din meniul principal EDIT;
- se salvează reprezentarea grafică.

30. CREAREA INTERFEŢELOR GRAFICE INTERACTIVE

Crearea interfeţelor grafice interactive (Graphics User Interface) se face în MATLAB cu următoarele funcţii:

ginput Funcţie pentru citirea datelor de pe grafic cu mouse-ul;
uicontrol Creează elemente de control pentru interfeţe grafice;
uimenu Creează meniuri pentru interfeţe grafice;
uiputfile Utilizează „ferestra de dialog" pentru a stabili numele fişierului care se salvează;
uigetfile Utilizează „ferestra de dialog" pentru a stabili numele fişierului care se încarcă;
uisetcolor Utilizează „ferestra de dialog" pentru a stabili matricea de culoare;
uisetfont Utilizează „ferestra de dialog" pentru a stabili fonturile.

30.1. Crearea elementelor de control pentru interfeţele grafice interactive

MATLAB foloseşte programarea cu obiecte pentru controlul interactiv al reprezentărilor grafice. Pentru aceasta sunt predefinite şase elemente de control interactiv:

1. butoane prin apăsare (push button);
2. butoane radio (radio button);
3. potenţiometre liniare (sliders);
4. listă de opţiuni (popup menu);
5. casete de editare (editable text);
6. casete de control (chek box).

1. Butoanele prin apăsare sunt comutatoare care generează o acţiune dacă sunt apăsate, însă nu rămân în starea apăsat. Pentru activarea unui buton prin apăsare se apasă şi se eliberează butonul mouse-ului. Aceste butoane se utilizează dacă acţiunea pe care o realizează nu este dependentă de nici o altă acţiune executabilă prin interfaţă.

2. Butoanele radio sunt mutual exclusive (numai unul poate fi în starea apăsat în acelaşi timp). Pentru activarea butonului radio se apasă şi se eliberează butonul mouse-ului. Starea butonului este indicată prin afişare. Prin algoritm trebuie ca implementarea acestor butoane să fie mutual exclusivă.

3. Potenţiometrele liniare acceptă valori numerice la intrare şi permit utilizatorului să deplaseze cursorul (prin apăsarea butonului mouse-lui, fără eliberarea acestuia) în interiorul unor limite. Poziţia cursorului indică valoarea numerică selectată la eliberarea butonului mouse-ului. Se pot stabili valorile minimă, maximă şi iniţială.

4. Lista de opţiuni a meniului este desfăşurată în momentul selectării. Dacă nu este activată, se afişează un singur buton cu un text care reprezintă configuraţia curentă. Această posibilitate de control grafic se foloseşte atunci când se doreşte selectarea unei opţiuni dintr-o listă mutual exclusivă, fără modificarea valorilor ca în cazul butoanelor radio.

5. Casetele de editare permit introducerea unui text de la tastatură. După ce acesta a fost scris în casetă, se apasă tasta „ENTER" sau se selectează butonul care lansează execuţia. Se utilizează în cazul în care programul trebuie să preia interactiv un text.

6. Casetele de control sunt generate prin acţiunea de apăsare, însă rămân în această stare până la o nouă selectare. Aceste obiecte sunt utilizate pentru a permite folosirea unui număr de opţiuni independente, basculând între două stări. Pentru activarea casetei de control se apasă şi se eliberează butonul mouse-lui; starea este indicată prin afişare.

Pentru crearea elementelor de control enumerate mai sus, folosite de interfeţele grafice interactive, se foloseşte funcţia *uicontrol*; se apelează cu sintaxa:

h=*uicontrol*('Nume_proprietate', Valoare_proprietate, ...)

Interfeţele grafice sunt copii ai obiectului figură şi sunt independente de obiectul axe. Funcţia *uicontrol* acceptă perechile proprietate/valoare ca argumente de intrare. Proprietăţile acestui obiect sunt descrise mai jos şi pot fi stabilite sau afişate cu funcţiile *set* şi *get*.

30.1.1. Proprietăţile elementelor de control ale interfeţelor grafice interactive

Proprietăţile elementelor de control pentru interfeţe grafice interactive pot fi specificate la creare prin includerea perechilor nume/ valoare_proprietate sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi sunt (valorile implicite sunt cele subliniate) :

1. `BackgroundColor` Culoarea specificată

Această proprietate specifică culoarea folosită pentru a colora fondul dreptunghiului definit prin interfaţa de control. Culoarea poate fi specificată printr-un triplet RGB sau prin numele de culori predefinite în MATLAB.

2. `CallBack` şir de caractere

Această proprietate permite evaluarea oricărei expresii MATLAB (inclusiv nume de fişiere-M sau funcţii) conţinută în şirul de caractere la activarea obiectului interfaţă.

3. `Children` vector al identificatorilor

Obiectul interfaţă nu are copii, astfel încât această proprietate este întotdeauna o matrice goală.

4. `ForegroundColor` Culoare specificată

Această proprietate precizează culoarea textului afişat în obiectul interfaţă. Culoarea poate fi specificată printr-un triplet RGB sau prin numele de culori predefinite în MATLAB.

5. `HorizontalAlignment` left | center | right

Această proprietate specifică modul de aliniere a textului definit de proprietatea `String` în obiectul interfaţă definit.

6. `Max` scalar

Această proprietate specifică cea mai mare valoare alocată proprietăţii `Value`. Pentru butoane radio şi casete de control, care operează cu comutatoare „on/off", aceasta reprezintă setarea lui `Value` când interfaţa este în poziţia „on". Pentru liste de opţiuni, această proprietate specifică numărul maxim de opţiuni care pot fi definite de meniu. Pentru potenţiometre liniare, această proprietate este cea mai mare valoare care poate fi selectată. Valoare implicită este 1.

7. `Min` scalar

Această proprietate defineşte cea mai mică valoare alocată proprietăţii `Value`.

Pentru butoane radio şi casete de control, care operează cu comutatoare „on/off", aceasta reprezintă setarea lui `Value`, când interfaţa este în poziţia „off". Pentru liste de opţiuni, această proprietate specifică numărul minim de opţiuni care pot fi definite de meniu. Pentru potenţiometre liniare, această proprietate este cea mai mică valoare care poate fi selectată. Valoare implicită este 0.

8. `Parent` identificator

Această proprietate returnează identificatorul obiectului figură (părinte), în care este afişat obiectul interfaţă. Funcţia *gca* returnează, de asemenea, identificatorul axelor curente ale figurii. Această proprietate se poate seta prin specificarea identificatorului obiectului părinte ca prim argument al funcţiei *uicontrol* (fără a fi necesar cuvântul „Parent"). Spre exemplu:

uicontrol(1, `'Style','slider'`,...)

determină identificatorul figurii curente.

9. `Position` vector [4 elemente]

Această proprietate specifică mărimea şi localizarea obiectului interfaţă în fereastra grafică, definită prin vectorul:

poz=[orizontal, vertical, lungimea, lăţimea]

unde :

- orizontal şi vertical sunt distanţele, pe orizontală şi verticală, de la colţul din stânga-jos al ferestrei grafice până la colţul din stânga-jos al dreptunghiului care defineşte obiectul interfaţă;
- lungimea şi înălţimea sunt dimensiunile dreptunghiului care defineşte obiectul interfaţă.

Elementele vectorului poz sunt date în unităţile de măsură specificate de proprietatea `Units`. MATLAB-ul utilizează implicit unităţile normalizate, la care (0,0) este colţul din stânga-jos, iar (1,1) este colţul din dreapta-sus al ferestrei grafice.

10. `String` şir de caractere

Această proprietate defineşte eticheta butoanelor prin apăsare, a butoanelor radio, a casetelor de control şi a listelor de opţiuni. Pentru introducerea mai multor opţiuni într-o listă de meniuri, sau pentru a avea linii multiple într-o casetă de control, se introduc şirurile separate de caracterul „ | " (bară verticală). Întreaga listă trebuie plasată între ghilimele (nu fiecare şir de caractere). Pentru casetele de editare, aceasta proprietate este setată la şirul introdus de utilizator.

11. `Style` pushbutton | radiobutton | checkbox | slider | edit | popupmenu

Această proprietate defineşte tipul de obiect de control care se creează: buton prin apăsare, buton radio, casetă de control, potenţiometru liniar, casetă de editare sau listă de opţiuni.

12. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul interfaţă şirul de caractere este întotdeauna *„uicontrol"*.

13. `Units` pixels / normalized / inches / centimeters / points

Această proprietate specifică unităţile de măsură utilizate pentru interpretarea proprietăţii `Position`. În toate cazurile măsurarea se face din colţul stânga-jos al ferestrei grafice. Unităţile normalizate transformă colţul din stânga-jos al ferestrei grafice în coordonatele (0,0), iar colţul din dreapta-sus în (1.0, 1.0). Celelalte unităţi de măsură sunt unităţi absolute, iar unitatea punct (points) este egală cu 1/72 inch.

14. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului, care poate fi văzută folosind funcţia *get*.

15. `Value` scalar

Valorile posibile ale acestei proprietăţi depind de metoda de control:

- butoanele radio şi cutiile de căutare stabilesc „`Value`" la „Max" (uzual 1) dacă acestea sunt „on" (butonul apăsat) şi „Min" (uzual 0) dacă sunt „off" (butonul neapăsat);
- potenţiometrele liniare stabilesc „Value" la numărul indicat de rigla cursorului, raportat la intervalul dintre „Min" şi „Max";
- lista de opţiuni stabileşte proprietatea „Value" la indexul cuvântului selectat;
- butoanele prin apăsare şi casetele de editare nu stabilesc această proprietate.

Proprietatea `Value` se poate stabili fie interactiv, cu mouse-ul, fie prin intermediul funcţiei *set*. Afişarea pe ecran reflectă schimbările făcute prin stabilirea directă a proprietăţii `Value`.

16. `Visible` on/off

Această proprietate determină dacă obiectul de control al interfeţei este sau nu vizibil pe ecran.

Exemplul 30.1.1. Să se creeze un buton prin apăsare, care şterge la selectare reprezentările grafice din axele curente.

Soluţia presupune secvenţa MATLAB :

```
h=uicontrol('Style','Pushbutton',...
'Units','normalized','Position',...
[.1 .8 .3  .1], 'Callback',...
'cla','String','Clear')
```


Exemplul 30.1.2. Să se creeze o interfaţă interactivă care schimbă culoarea unei reprezentări grafice prin selectarea unei matrice de culoare dintr-o listă de meniuri (popup menu)

Secvenţa de program care rezolvă această problemă este:

```
hp=uicontrol('Style','Popup','String',...
'hsv|hot|cool|gray|','Units',...
'normalized','Position',...
[.1 .1 .3  .2], 'Callback',...
'setmap')
```

În acest caz obiectul defineşte o listă de patru culori, care pot fi alese din meniu: hsv, hot, cool şi gray. Această opţiune este specificată cu proprietatea `String` şi argumentele separate de caracterul „ | " (bară verticală). Selectarea uneia dintre aceste opţiuni apelează fişierul_M numit **setmap.m**, care poate avea conţinutul:

```
val=get(hp, 'Value');
if val==1
      colormap(hsv)
elseif val==2
      colormap(hot)
elseif val==3
      colormap(cool)
elseif val==4
      colormap(gray)
end
```

Proprietatea `Value` a acestui obiect (*uimenu*) conţine numărul care indică ce opţiune de culoare a fost selectată şi extrage din fişierul **setmap.m** matricea de culoare corespunzătoare.

Exemplul 30.1.3. Să se creeze toate cele şase tipuri de obiecte de control al interfeţelor grafice.

Cu secvenţa MATLAB:

```
hpb=uicontrol('Style','Pushbutton',...
     'Units','normalized','Position',...
     [.1 .8 .3  .1], 'Callback',...
     'cla','String','Buton prin apasare')
hrb=uicontrol('Style','Radiobutton',...
     'Units','normalized','Position',...
     [.6 .8 .3  .1], 'Callback',...
     'cla','String','Buton radio')
hsl=uicontrol('Style','Slider','String',...
     'Potentiometru liniar','Units',...
     'normalized','Position',...
     [.1 .5 .3 .1], 'Min', 100,'Max',500 ).
hpm=uicontrol('Style','Popup','String',...
     'hsv|hot|cool|gray|','Units',...
     'normalized','Position',...
     [.1 .2 .3  .1],'Callback',...
     'setmap')
hcb=uicontrol('Style','Checkbox','String',...
```

```
        'Cutie de control','Units',...
        'normalized','Position',...
        [.6 .5 .3 .1], 'Callback',...
        'figure')
hed=uicontrol('Style','Edit','String',...
        'Cutie de editare','Units',...
        'normalized','Position',[.6 .2 .3 .1])
```

se obţine fereastra grafică din figura 30.1, în care sunt reprezentate obiectele de control ale interfeţei.

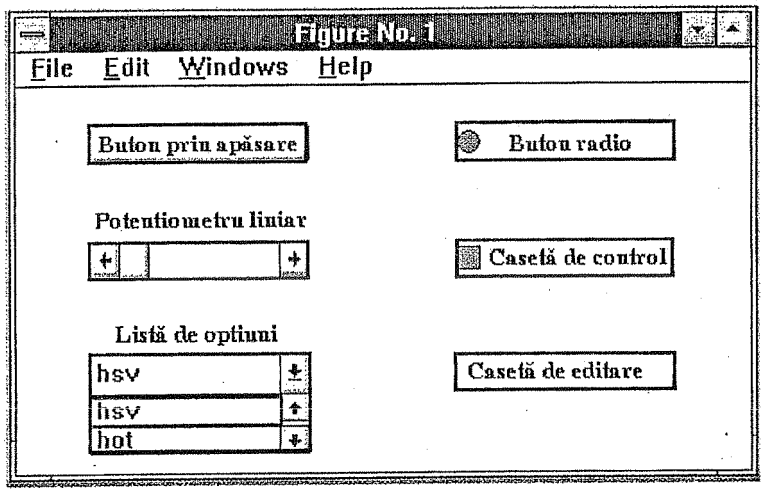


Fig. 30.1. Obiecte de control ale interfeţei grafice create cu funcţia *uimenu*.

30.2. Crearea meniurilor pentru interfeţele grafice

Pentru crearea unor meniuri cu acţiuni predefinite, care se afişează în bara superioară de meniuri, se utilizează funcţia *uimenu*; se apelează cu sintaxele:

h=*uimenu*('Nume_proprietate', Valoare_proprietate, ...)

hsub=*uimenu*(h, 'Nume_proprietate', Valoare_proprietate, ...)

Fiecare opţiune a meniului poate fi la rândul ei un meniu, care afişează un submeniu, atunci când este selectată. Submeniurile se crează prin apelarea funcţiei *uimenu* având ca prim argument identificatorul meniului principal. Dacă nu se specifică nici un identificator al unui meniu existent, se creează un meniu principal, care se plasează în bara superioară de meniuri (la nivelul cel mai înalt).

Funcţia *uimenu* acceptă perechile proprietate/valoare ca argumente de intrare. Proprietăţile acestui obiect sunt descrise în continuare şi pot fi stabilite sau afişate cu funcţiile *set* şi *get*.

30.2.1. Proprietăţile obiectului meniu

Proprietăţile obiectului meniu pot fi specificate la crearea acestula, prin includerea perechilor nume/valoare_proprietate sau ulterior, prin identificarea obiectului şi folosirea funcţiilor *set* şi *get*.

Lista proprietăţilor acceptate, precum şi valorile posibile ale fiecărei proprietăţi, sunt (valorile implicite sunt cele subliniate) :

1. `Accelerator` Caracter

Această proprietate specifică combinaţia de caractere echivalentă acţiunii de apelare a unei opţiuni din meniu. Acceleratorul permite utilizatorului să selecteze o opţiune din meniu prin tastarea combinaţiei de taste **<Ctrl>+<Caracter>**.

2. `CallBack` şir de caractere

Această proprietate permite evaluarea oricărei expresii MATLAB (inclusiv nume de fişiere-M sau funcţii) conţinută în şirul de caractere la activarea opţiunii din obiectul meniu.

3. `Children` vector al identificatorilor

Copiii obiectului meniu sunt alte meniuri (numite submeniuri). Identificatorul conţine câte un vector pentru fiecare submeniu.

4. `Enable` <u>on</u> | off

Această proprietate controlează o opţiune care se poate sau nu selecta. Dacă o opţiune nu se poate selecta, eticheta acesteia este scrisă cu litere de culoare deschisă (estompat).

5. `Label` Şir de caractere

Această proprietate specifică textul etichetelor opţiunilor meniului.

6. `Parent` identificator

Această proprietate returnează identificatorul obiectului părinte, care poate fi un obiect figură sau un obiect meniu. Pentru crearea submeniurilor trebuie setată această proprietate.

7. `Position` scalar

Această proprietate specifică localizarea obiectului meniu pe bara superioară, de la stânga la dreapta, funcţie de valoarea scalarului.

8. `Separator` on | off

Această proprietate permite separarea cu linii a fiecărei opţiuni a unui meniu. Dacă este stabilit pr „on", opţiunile din listă sunt separate cu linii.

9. `Type` Şir de caractere (numai pentru citire)

Această proprietate identifică tipul de obiect grafic. Pentru obiectul meniu şirul de caractere este întotdeauna „*uimenu*".

10. `UserData` matrice

Proprietatea `UserData` este orice matrice asociată obiectului, care poate fi văzută folosind funcţia *get*.

11. `Visible` on/off
Această proprietate determină dacă obiectul meniu este sau nu vizibil pe ecran.

Exemplul 30.2.1. Să se creeze un meniu principal cu eticheta „Aplicaţii" şi cu o listă cu trei opţiuni: New, Save şi Quit. New are un submeniu cu două opţiuni: New_Figure şi New_M.
Cu secvenţa MATLAB:

```
f=uimenu('Label','Aplicatii');
n=uimenu(f,'Label','New')
uimenu(n,'Label','New_Figure','CallBack','figure');
uimenu(n,'Label','New_M','CallBack','load');
uimenu(f,'Label','Save','CallBack','save');
uimenu(f,'Label','Quit','CallBack','exit',...
'Separator','on','Accelerator','Q');
```

se obţine meniul prezentat în figura 30.2.

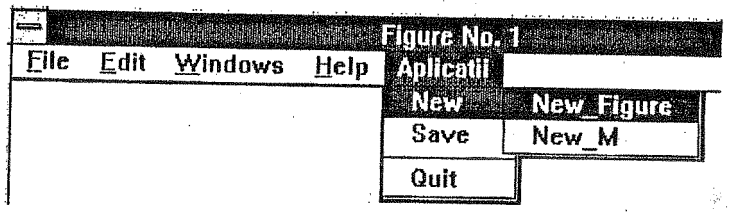


Fig. 30.2. Crearea unei interfeţe grafice cu meniu şi submeniuri.

30.3. Citirea interactivă a datelor de pe grafice

Pentru introducerea interactivă (cu mouse-ul) a datelor de pe grafice reprezentate pe ecran se foloseşte funcţia *ginput*, se apelează cu una dintre sintaxele:

[x,y]=*ginput*(N) - returnează N puncte de pe figura curentă, ale căror coordonate sunt vectorii x şi y. Pentru fiecare dată se poziţionează cursorul mouse-lui în punctul dorit şi se apasă butonul acestuia pentru confirmare. La terminarea introducerii celor N puncte sunt returnaţi vectorii x şi y;

[x,y]=*ginput* - permite introducerea unui număr neprecizat de puncte, operaţia încheindu-se prin apăsarea tastei „ENTER";

[x,y,B]=ginput(N) - returnează coordonatele celor N date selectate în vectorii x şi y, precum şi variabila B care reprezintă butonul cu care a fost selectată data (1, 2 sau 3, de la stânga la dreapta). Pentru mouse-urile cu două butoane, B are valorile 1 sau 3.

Punctele care se doresc introduse interactiv într-un fişier de date se selectează cu săgeata mouse-ului. După aceea, variabila în care au fost returnate aceste valori se salvează cu funcţia *save*, ca orice variabilă a mediului de lucru MATLAB.

30.4. Selectarea interactivă a fişierelor citite

Pentru selectarea interactivă a numelui şi locului de unde se citesc date sau programe MATLAB, se utilizează funcţia *uigetfile*, care afişează o „casetă de dialog"; se apelează cu sintaxa:

[nume, cale]=*uigetfile*('filtrul', 'titlul_casetei',X,Y)

- parametrul „filtrul" determină care fişiere sunt iniţial afişate în fereastra de dialog. Spre exemplu, dacă acesta este „*.m", sunt listate toate fişierele MATLAB cu extensia *.m;
- parametrul „titlul_casetei" este un şir de caractere care se afişează ca titlu al casetei de dialog;
- parametrii X şi Y definesc poziţia iniţială a ei de dialog, în unităţi pixel (nu toate sistemele acceptă această opţiune);
- parametrul de ieşire „nume" este un şir de caractere în care este returnat numele fişierului selectat din „caseta de dialog". Dacă este întâlnită o eroare sau este selectat butonul **Cancel**, acesta este setat la 0;
- parametrul de ieşire „cale" este un şir de caractere în care este returnată calea fişierului selectat din caseta de dialog. Dacă este întâlnită o eroare sau este selectat butonul **Cancel**, acesta este setat la 0.

Dacă fişierul menţionat nu există, se afişează un mesaj de eroare şi controlul este returnat „casetei de dialog". Utilizatorul poate introduce alt nume de fişier sau poate selecta butonul **Cancel**. Toţi parametrii funcţiei sunt opţionali, însă dacă unul este utilizat, toţi parametrii anteriori trebuie utilizaţi.

Exemplul 30.4.1. Să se folosească o „casetă de dialog" pentru selectarea fişierelor cu extensia „*.m".
Cu secvenţa MATLAB:

```
[num,cal]=uigetfile('*.m','SELECTEAZĂ')
```

se obţine „caseta de dialog" din figura 30.3.

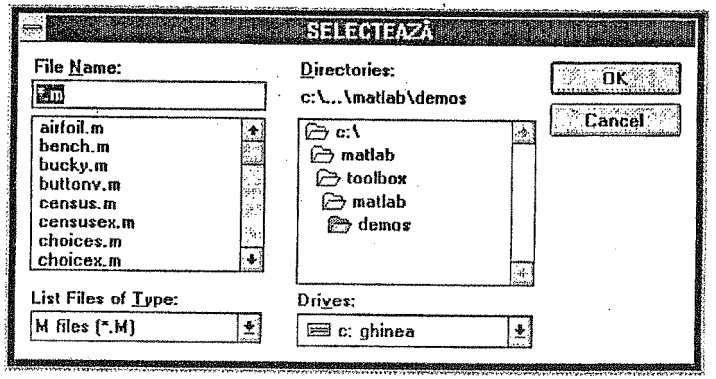


Fig. 30.3. Caseta de dialog pentru selectarea interactivă a fişierelor citite.

30.5. Stabilirea interactivă a fişierelor salvate

Pentru stabilirea interactivă a numelui şi locului unde se vor salva date sau programe MATLAB se utilizează funcţia *uiputfile*, care afişează o „casetă de dialog"; se apelează cu sintaxa:

[nume, cale]=*uipufile*('fişier_iniţial', 'titlul_casetei',X,Y)

- parametrul „fişier_iniţial" precizează numele fişierului implicit în care se salvează datele. Spre exemplu, dacă acesta este „test.m", sunt listate toate fişierele cu MATLAB cu extensia *.m;
- parametrul 'titlul_casetei' este un şir de caractere care se afişează ca titlu al cutiei de dialog;
- parametrii X şi Y definesc poziţia iniţială a cutiei de dialog, în unităţi pixel (nu toate sistemele acceptă această opţiune);
- parametrul de ieşire „nume" este un şir de caractere în care este returnat numele fişierului selectat din „caseta de dialog". Dacă este întâlnită o eroare sau este selectat butonul Cancel, acesta este setat la 0;
- parametrul de ieşire „cale" este un şir de caractere în care este returnată calea fişierului selectat din caseta de dialog. Dacă este întâlnită o eroare sau este selectat butonul **Cancel**, acesta este stabilit la 0.

Dacă fişierul există, suprascrierea acestuia cu date noi se face numai dacă utilizatorul acceptă, prin selectarea butonului „**YES**", concatenată cu o comandă de salvare.

Toţi parametrii funcţiei sunt opţionali, însă dacă unul este utilizat, toţi parametrii anteriori trebuie utilizaţi.

Exemplul 30.5.1. Să se folosească o „casetă de dialog" pentru salvarea fişierelor, afişând iniţial pe cele cu extensia „*.mat".
Cu secvenţa MATLAB:

```
[num,cal]=uiputfile('*.mat','Save As')
```

se obţine „caseta de dialog" din figura 30.4.

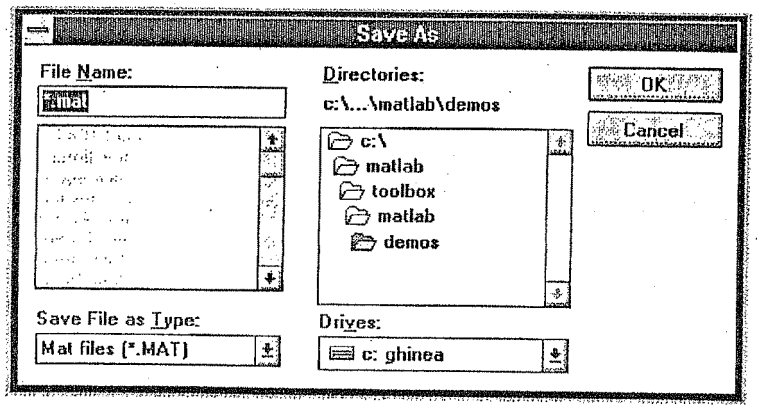


Fig. 30.4. Caseta de dialog pentru stabilirea interactivă a fişierelor salvate.

30.6. Selectarea interactivă a fonturilor

Pentru selectarea interactivă a fonturilor utilizate de o aplicaţie MATLAB se utilizează funcţia *uisetfont*, care afişează o „casetă de dialog" cu numele tuturor fonturilor gestionate de Windows; se apelează cu sintaxa:

H=*uisetfont*(HIN, 'titlul_casetei')

Această funcţie se utilizează la selectarea fonturilor ca intrări pentru obiectele grafice.

Parametrii sunt opţionali şi pot fi specificaţi în orice ordine. Dacă HIN este utilizat, trebuie să se specifice identificatorul obiectului grafic text sau axe. Proprietăţile curente ale fonturilor asignate acestui obiect sunt utilizate pentru a iniţializa „caseta de dialog" a fonturilor.

Dacă parametrul „titlul_casetei" este utilizat, acesta este afişat ca nume al casetei de dialog, iar argumentul de ieşire H este identificator al obiectului grafic. Dacă HIN este specificat, H este identic cu HIN, iar dacă nu este specificat se creează un nou obiect text prin selectarea proprietăţilor fonturilor şi se returnează identificatorul acestuia.

Dacă este întâlnită o eroare sau este selectat butonul Cancel, identificatorul H este setat la 0.

Exemplul 30.6.1. Să se afişeze o casetă de dialog care să permită selectarea interactivă a fonturilor folosite la scrierea axelor unei reprezentări grafice.
Cu secvenţa MATLAB:

```
uisetfont(gca,'Fonturi')
```

pe ecran apare caseta de dialog din figura 30.5, cu titlul „Fonturi", care permite utilizatorului stabilirea interactivă a tipurilor de fonturi, a dimensiunilor acestora, precum şi a altor caracteristici.

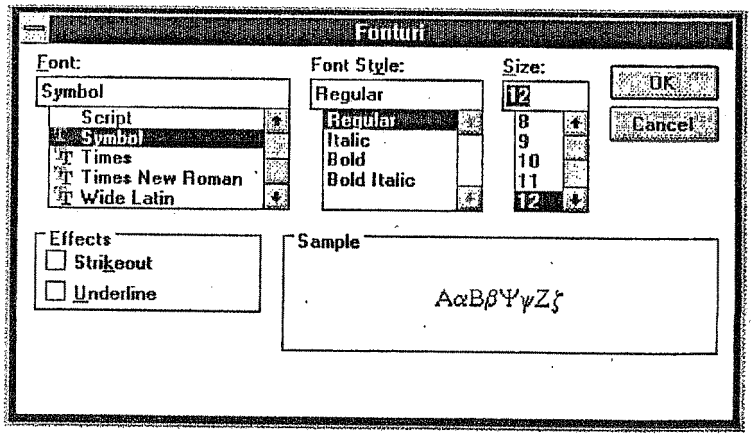


Fig. 30.5. Caseta de dialog pentru selectarea interactivă a fonturilor.

30.7. Selectarea interactivă a culorilor

Pentru alegerea interactivă a culorilor utilizate de o aplicaţie MATLAB se utilizează funcţia *uisetcolor* care afişează o casetă de dialog cu numele specificat şi cu toate culorile gestionate de Windows; se apelează cu sintaxa:

C=*uisetcolor*(ARG, 'titlul_casetei')

şi se utilizează la selectarea culorilor ca intrări pentru obiectele grafice.
Parametrii sunt opţionali şi pot fi specificaţi în orice ordine. Parametrul ARG este fie un identificator al unui obiect grafic, fie un triplet RGB (spre exemplu [1 0 0] pentru roşu). În ambele cazuri, culoarea specificată este utilizată la iniţializarea casetei de dialog. Dacă nu este utilizată iniţializarea, culoarea folosită este negru.

Dacă parametrul „titlul_casetei" este utilizat, acesta este afişat ca nume al casetei de dialog, iar argumentul de ieşire C este tripletul RGB selectat. Dacă parametrul de intrare este identificator, culoarea obiectului grafic este stabilită la culoarea RGB selectată.

Dacă este întâlnită o eroare sau este selectat butonul Cancel, valoarea de ieşire a tripletului RGB este stabilită la 0.

Exemplul 30.7.1. Să se afişeze o casetă de dialog care să permită alegerea interactivă a culorilor folosite la fondul ferestrei grafice (obiectul figură) şi la fondul axelor (obiectul axe).
Cu secvenţa MATLAB:

```
Ca=uisetcolor(gca,'Fondul axelor')
Cf=uisetcolor(gcf,'Fondul figurii')
```

pe ecran apare caseta de dialog din figura 30.6, cu titlul „Fondul axelor" / „Fondul figurii", care permite utilizatorului alegerea interactivă a culorii.

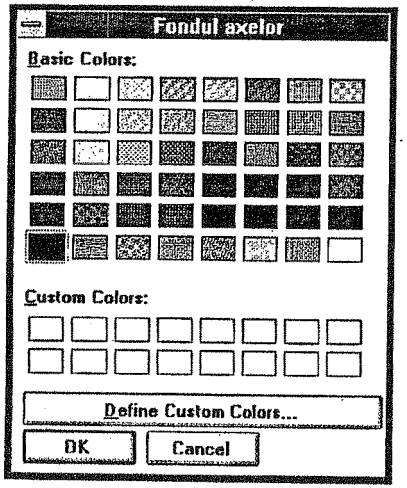


Fig. 30.6. Caseta de dialog pentru selectarea interactivă a culorilor

Partea a IV-a
31. APLICAŢII

31.1. Semnale cu ecouri

Aplicaţia dezvoltă un program MATLAB pentru adăugarea de ecouri la un semnal digital original. Se defineşte mai întâi semnalul original, precum şi întârzierile şi factorii de atenuare ai ecourilor. Semnalul digital este un vector, dacă conţine numai valorile eşantioanelor semnalului analogic, sau o matrice dacă conţine şi momentele de eşantionare.

Să se scrie un program MATLAB pentru generarea unui semnal cu trei ecouri, conform diagramei prezentate în figura 31.1. Pentru amplificare s-a folosit semnul grafic triunghi, iar pentru celulele de întârziere, dreptunghiul. Factorii de amplificare A şi întârzierile Δt sunt înscrise pe figură.

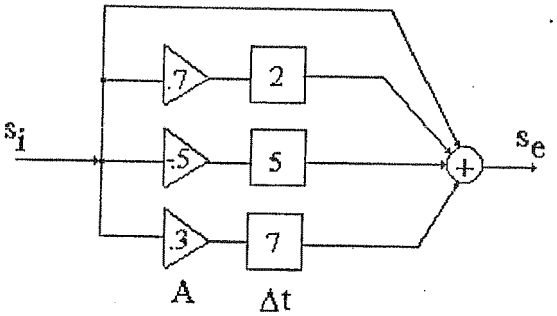


Figura 31.1. Generarea ecourilor unui semnal s_i.

Semnalul de ieşire s_e este rezultatul suprapunerii a trei ecouri peste semnalul de intrare s_i, definit de următorul set de valori:

Timp	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9 ...10
Valoare	0.0	0.5	1.0	0.5	0.0	-0.5	-1.0	-0.5	0.0	0

Se consideră o înregistrare pe o durată de 10 secunde, cu o perioadă de eşantionare de 0.1 secunde. Să se reprezinte grafic semnalul cu ecouri.

Programul MATLAB este următorul:

```
t=0:0.1:10;
si=zeros (t);
so=[0 0.5 1 0.5 0 -0.5 -1 -0.5 0];
si(1:9)=so; %semnalul original
```

```
ecou_1 = zeros(t)
ecou_1(21:29)=0.7*so;
ecou_2 = zeros(t)
ecou_2(51:59)=-0.5*so
ecou_3= zeros(t)
ecou_3(71:79)=0.3*so
se=si+ecou_1+ecou_2 + ecou_3
save ecou t se -ascii
plot(t,se); grid
title('Semnal cu ecouri')
xlabel('t[secunde]'); ylabel('semnal')
```

Semnalul rezultat este reprezentat în figura 31.2.

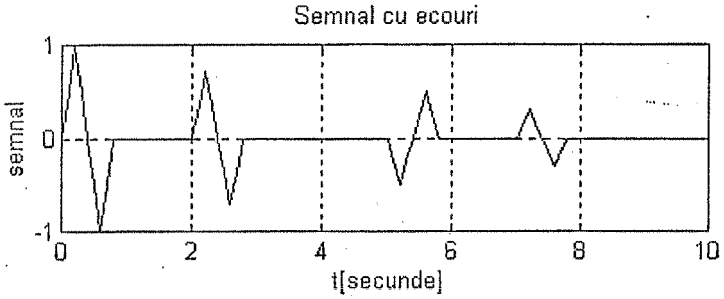


Figura 31. 2.

31.2. Semnale aleatoare cu distribuţie uniformă

Numerele aleatoare generate cu funcţia *rand* din MATLAB sunt uniform distribuite în intervalul [0,1]. Transformarea numerelor r distribuite uniform pe intervalul [0,1] în numerele x distribuite uniform pe intervalul [a, b], se realizează cu relaţia :

x=(b-a)r+a

Înălţimea dreptunghiului de reprezentare a numerelor aleatoare, în cazul când acestea sunt uniform distribuite pe intervalul [a,b], este 1/(b-a). Spre exemplu, dacă numerele sunt distribuite uniform în intervalul [-5, 5], înălţimea va fi 1/10, iar relaţia de conversie este x=10r-5.

Cu secvenţa MATLAB:

```
r=rand(1,5000);
a=2; b=6;
x=(b-a)*r+a;
[n,xg]=hist(x,20);
yn=n/max(n)/(b-a);
bar(xg,yn)
```

se obţine reprezentarea grafică din figura 31.3.

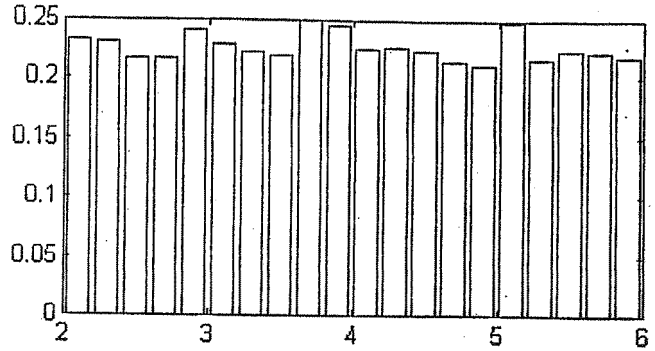


Fig. 31.3. Funcţia densitate de probabilitate uniformă pentru a=2, b=6.

O secvenţă de numere aleatoare poate fi de exemplu un semnal aleator. Unul dintre parametrii importanţi care caracterizează acest semnal, este valoarea medie calculată în MATLAB cu funcţia *mean*. Dacă numărul de eşantioane tinde către infinit, valoarea medie a unui semnal aleator cu distribuţie uniformă este egală cu valoarea medianei funcţiei densitate de probabilitate, adică:

$$\mu = \frac{X_{min} + X_{max}}{2}$$

unde X_{min} şi X_{max} sunt valorile minimă şi maximă ale semnalului.

Varianţa unui semnal aleator se calculează în MATLAB prin ridicarea la pătrat a valorii abaterii standard - funcţia *std*. În teoria probabilităţilor se arată că varianţa este media valorilor pătrate din care se scade valoarea medie :

$$\sigma^2 = \frac{\sum_{k=1}^{N} x_k^2}{N} - \mu^2$$

Varianţa distribuţiei uniforme are expresia

$$\sigma^2 = \frac{(X_{max} - X_{min})^2}{12}$$

Penultima relaţie se aplică oricărei secvenţe de numere aleatoare, în timp ce ultima relaţie este valabilă numai pentru numere aleatoare cu distribuţie uniformă. În cazul eşantionării unui semnal, valorile mediei şi varianţei nu corespund exact cu rezultatele date de expresiile de mai sus; cu cât se utilizează mai multe eşantioane, cu atât valorile calculate se apropie mai mult de valorile analitice.

Se generează un semnal aleator cu distribuţie uniformă în intervalul [-5,5]. Valoarea medie analitică a acestui semnal este 0, iar varianţa sa 100/12=8,333. Să se calculeze cu funcţii MATLAB media şi varianţa semnalului aleator cu 10, 100, 1000 şi 50000 eşantioane.

Secvenţa MATLAB:

```
n=[10  100  1000  5000];
a=-5; b=5;
m=length (n);
```

```
for i = 1:m
        S=rand(1,n(i));
        S=(b-a)*S+a;
        M(i,1)=mean(S);
        V(i,1)=std(S)^2;
end
nMV=[n'  M  V]
disp('Esantioane  Media  Varianta')
disp(nMV)
```

returnează rezultatele:

Eşantioane	Media	Varianţa
10	-0.3223	5.0953
100	0.1009	8.5951
1000	-0.0646	8.4422
5000	-0.0205	8.3141

Se observă că, prin creşterea numărului de eşantioane, valorile numerice ale mediei şi varianţei se apropie de valorile analitice.

31.3. Semnale aleatoare cu distribuţie normală

Există fenomene în al căror studiu statistic sunt necesare secvenţe de numere aleatoare cu probabilitate de apariţie de tip distribuţie normală. Repartiţia normală este cunoscută sub denumirea de distribuţie Gauss.

Funcţia densitate de probabilitate Gaussiană are expresia:

$$f(x)=\frac{1}{\sigma^2\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

unde μ este media şi σ^2 este varianţa.

Valoarea medie aleatoare corespunde coordonatei x pentru care funcţia densitate are un maxim. În repartiţia normală probabilitatea de apariţie a unei valori apropiată de valoarea medie este mai mare decât probabilitatea de apariţie a unei valori îndepărtată de medie.

Cu cât varianţa unei secvenţe de numere aleatoare este mai mică, cu atât probabilitatea ca numerele generate să fie mai aproape de valoarea medie este mai mare. Se demonstrează că 68,26% din valori se găsesc în intervalul $[\mu-\sigma, \mu+\sigma]$, 95,45% în intervalul $[\mu-2\sigma, \mu+2\sigma]$ şi 99,74% în intervalul $[\mu-3\sigma, \mu+3\sigma]$.

Să se scrie un program utilizând funcţia *randn* pentru generarea unei secvenţe de numere aleatoare cu distribuţie normală, caracterizate prin varianţa 3 şi media 5.

Deoarece funcţia *randn* returnează secvenţe cu media 0 şi varianţa 1, se aplică transformarea:

x=a*r+b

Secvenţa următoare: generează 10000 numere aleatoare cu distribuţie normală:

```
media=5; var=3; n=10000;
```



```
x=sqrt(var)*randn(1,n)+media;
[n,y]=hist(x,25);plot(y,n); grid
```

generează 10000 numere aleatoare, a căror distribuţie este reprezentată în figura 31.4.

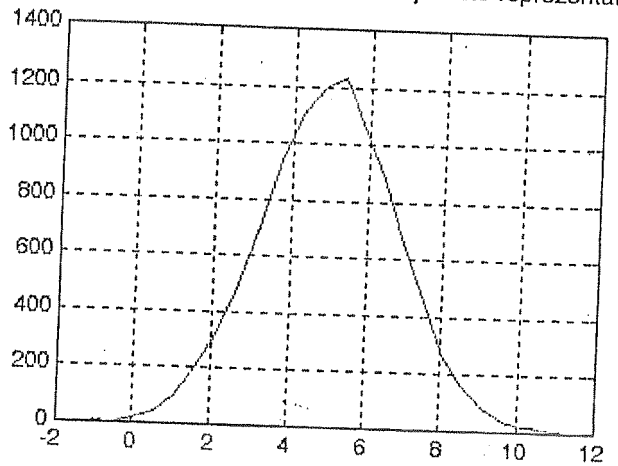


Fig. 31.4

31.4. Raportul semnal - zgomot

Pentru testarea unor echipamente electronice se utilizează semnale de intrare care conţin o funcţie armonică cu zgomot aditiv. Raportul semnal-zgomot, RSZ, este definit cu ajutorul conceptului de putere a semnalului; puterea unui semnal reprezentat prin vectorul x este estimată prin medierea sumei pătratelor valorilor eşantioanelor:

$$P\approx\frac{\sum_{k=1}^{N}x_k^2}{N}$$

Puterea semnalului poate fi exprimată şi în funcţie de varianţă şi pătratul mediei, cu relaţia:

$$P=\sigma^2+\mu^2$$

Raportul semnal - zgomot se calculează cu relaţia:

$$RSZ=\frac{P_{semnal}}{P_{zgomot}}$$

unde P_{semnal} este puterea semnalului, iar P_{zgomot} este puterea zgomotului. Se exprimă, de regulă, în decibeli (dB) prin relaţia:

$$RSZ_{dB}=10\log\left(\frac{P_{semnal}}{P_{zgomot}}\right)$$

Să se scrie un program în MATLAB care să reprezinte grafic un semnal

sinusoidal cu amplitudinea A=2V, frecvenţa f=100Hz, faza iniţială φ=0, eşantionat cu frecvenţa f_e=1000Hz, peste care se suprapune un zgomot uniform distribuit între [-0.25, 0.25]V. Să se calculeze raportul semnal-zgomot al acestui semnal.

Cu secvenţa MATLAB:

```
fe=1000; f=100; a=0.25; N=200; A=2;
RSZdB=10*log10(((A^2)/2)/((2*a)^2/12))
t=(0:N-1)*1/fe
s=A*sin(2*pi*f*t)+(2*a*rand(size(t))-a);
plot(t,s)
title ('Sinusoida cu zgomot')
xlabel('t [secunde]')
ylabel('Amplitudinea[V]')
grid
```

se obţine RSZ= 19.8227 dB.

31.5. Curenţii într-un circuit electric

Fie circuitul din figura 31.6. în care sunt cunoscute valorile rezistenţei rezistoarelor şi tensiunea sursei.

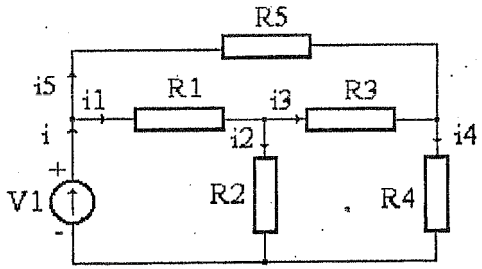


Figura 31.6.

Ansamblul ecuaţiilor valorilor necunoscute ale curenţilor:

$$\begin{cases} -I1 & & & & -I5 & +I6 & = & 0 \\ I1 & -I2 & -I3 & & & & = & 0 \\ & & I3 & -I4 & +I5 & & = & 0 \\ R1.I1 & & +R3.I3 & & -R5.I5 & & = & -V2 \\ R1.I1 & +R2.I2 & & & & & = & V1 \\ & -R2.I2 & +R3.I3 & +R4.I4 & & & = & 0 \end{cases}$$

se poate scrie sub forma matriceală RI=V, unde R este matricea coeficienţilor, I matricea curenţilor prin laturi, iar V matricea termenilor liberi:

$$R=\begin{bmatrix} -1 & 0 & 0 & 0 & -1 & 1 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ R1 & 0 & R3 & 0 & -R5 & 0 \\ R1 & R2 & 0 & 0 & 0 & 0 \\ 0 & -R2 & R3 & R4 & 0 & 0 \end{bmatrix} \quad I=\begin{bmatrix} I1 \\ I2 \\ I3 \\ I4 \\ I5 \\ I6 \end{bmatrix} \quad V=\begin{bmatrix} 0 \\ 0 \\ 0 \\ -V2 \\ V1 \\ 0 \end{bmatrix}$$

O variantă de program, care permite introducerea interactivă de la tastatură a valorilor rezistenţelor, precum şi a valorilor surselor de tensiune, este următoarea:

```
R=input('Introduceti val. rezistentelor [R1...5]:')
V=input('Introduceti valorile tensiunilor[V1,V2]:')
%Se initializeaza matricele
R=[-1,  0,  0,  0,  -1,  1;
    1,  -1,  -1,  0,  0,  0;
    0,  0,  1,  -1,  1,  0;
    R1,  0,  R3,  0,  -R5,  0;
    R1, R2,  0,  0,  0,  0;
    0,  -R2,  R3,  R4,  0,  0];
V=[0; 0; 0; -V2; V1; 0];
I=R\V
fprintf('Curentii prin laturile circuitului sunt :\n')
disp (I')
```

Acest program trebuie mai întâi scris într-un fişier M. La lansarea fişierului se afişează mesajele de introducere a valorilor rezistenţelor şi tensiunii sursei; se introduc valori numerice în ordinea indicilor, separate de caracterul virgulă, cu paranteze drepte la început şi sfârşit.

Pentru [R1...R5]=[1,1,1,1,1] şi [V1 V2]=[5,5], rezultă curenţii prin laturile circuitului:

1.2893 1.8553 -0.5660 1.3522 1.9182 3.2075

31.6. Circuite electrice în regim tranzitoriu

Un condensator de capacitate C încărcat la tensiunea U_0 este cuplat la bornele unui circuit care cuprinde un rezistor de rezistenţă R în serie cu o bobină având inductivitatea L. Să se scrie un program pentru calculul şi reprezentarea grafică a variaţiei în timp a intensităţii curentului în circuit. În cazul unui regim oscilant amortizat să se verifice soluţia prin compararea energiei iniţiale a condensatorului cu puterea dezvoltată prin efect Joule în rezistor.

Funcţia de variaţie în timp a tensiunii la bornele condensatorului satisface ecuaţia diferenţială de ordinul doi:

$$LC\frac{d^2u_c}{dt^2}+RC\frac{du_c}{dt}+u_c=0$$

cu condiţiile iniţiale:

$$uc(0)=U_0\,,\quad \left.\frac{duc}{dt}\right|_{t=0}=0$$

Curentul în circuit se determină cu relaţia:

$$i=-C\frac{du_c}{dt}$$

Se disting următoarele cazuri:

1. regim aperiodic, pentru $R>2\sqrt{L/C}$;
2. regim aperiodic critic, pentru $R=2\sqrt{L/C}$;
3. regim oscilant amortizat, pentru $R<2\sqrt{L/C}$;
4. regim oscilant neamortizat, pentru R=0.

În cazurile 3 şi 4 pulsaţia curentului are expresia:

$$\omega=\omega_0\sqrt{1-\left(\frac{R}{2\sqrt{1/C}}\right)^2}\,,\qquad \text{unde}\quad \omega_0=\frac{1}{\sqrt{LC}}$$

Aplicaţie numerică: U_0=100 V, C=100 μF, L=10 mH, R=3.5 Ω.

Pentru integrarea ecuaţiei diferenţiale se scrie următorul fişier funcţie-M:

```
function ucond=c_tranz(t,uc)
L=10*10^(-3);  C=100*10^(-6); R=3.5 ;
ucond(1)=(-R/L).*uc(1)-1/(L*C).*uc(2);
ucond(2)=uc(1);
```

înregistrat cu numele **c_tranz.m**.

Calculul variaţiei în timp a curentului în circuit se face cu secvenţa:

```
U0=100;
i0=0;
uc0=[ U0  i0 ];
L=10*10^(-3);  C=100*10^(-6); R=3.5 ;
omega0=1/sqrt(L*C);
omega=omega0*sqrt(1-(R/2/sqrt(L/C))^2);
T=2*pi/omega;
t0=0; tf=5*T;
[t, uc_i]=ode45('c_tranz', t0, tf, uc0);
uc=uc_i(:,1); %tensiunea pe condensator
i=-C*diff(uc)./diff(t); %curentul prin circuit
n=length(i);
plot(t(1:n),i); grid
end
```

Rezultatul programului este reprezentat grafic în figura 31.7.

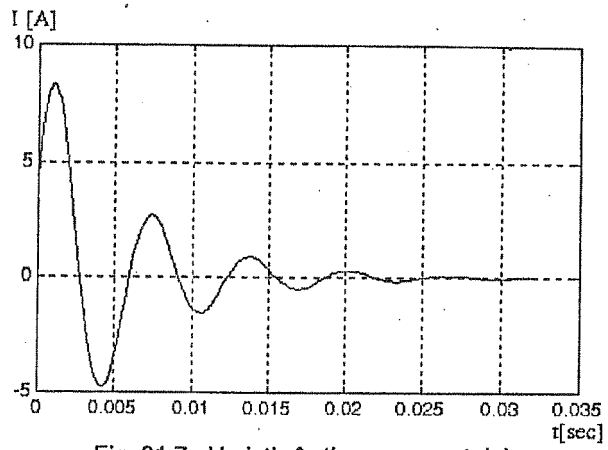


Fig. 31.7. Variaţia în timp a curentului.

31.7. Simularea numerică a fiabilităţii

Fiabilitatea unui sistem poate fi evaluată dacă sunt cunoscute prin studii statistice şi de probabilitate fiabilităţile componentelor acestuia. Pentru simularea unui studiu de fiabilitate trebuie precizate dacă componetele sunt conectate într-o schemă structurală de fiabilitate serie (figura 31.8a), sau paralel (figura 31.8b). Pentru ca informaţia să treacă de la A la B, în schema serie trebuie ca toate componentele să fie în stare de funcţionare, în timp ce într-o schemă structurală de fiabilitate paralel trebuie ca numai o singură componentă să fie în stare de funcţiune.

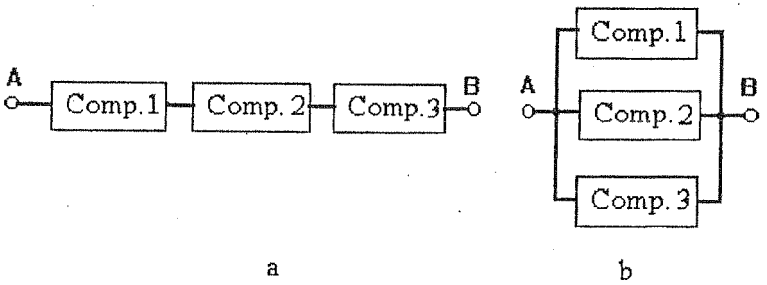


Fig. 31.8. Scheme structurale de fiabilitate. a) Serie. b) Paralel.

Simularea numerică a fiabilităţii unui sistem se face după următorul algoritm:

- pentru n componente conectate în serie, fiecare cu fiabilitatea p, se generează n numere uniform distribuite între zero şi unu. Dacă toate cele n numere sunt mai mici sau egale cu p, se apreciază că sistemul funcţionează;
- pentru n componente conectate în paralel, fiecare cu fiabilitatea p, se generează n numere aleatoare uniform distribuite între zero şi unu. Dacă cel puţin unul dintre

cele n numere este mai mare sau egal cu p, se apreciază că sistemul funcţionează.

Repetând de sute sau mii de ori această simulare numerică se estimează probabilitatea ca sistemul fizic asociat să funcţioneze, egală cu raportul dintre numărul de încercări reuşite şi numărul total de încercări.

Să se scrie un program MATLAB care să analizeze comportarea:

1. sistemului serie din figura 31.8a, utilizând componente cu fiabilitatea p=0.8;
2. sistemului paralel din figura 31.8b, utilizând componente cu fiabilitatea p=0.8. Să se compare fiabilităţile calculate utilizând 500 şi 1000 simulări;
3. sistemului cu două componente serie, prima cu fiabilitatea 0.8, iar a doua cu fiabilitatea 0.92.

Programul de simulare numerică pentru pct. 1 este următorul:

```
n = [500  1000]; % nr. de simulari
p = 0.8; % fiabilitatea componentelor
m = 3; % nr. de componente serie
for j = 1:length(n)
k = 0;
      for i = 1:n(j)
            x = rand(1,m);
         if all(x<=p)
            k = k+1;
         else
         end
      end
F(1,j) = k/n(1,j) % fiabilitatea
end
```

Se obţin fiabilităţile [0.5200 0.5000].

Programul de simulare numerică pentru pct. 2 este:

```
n=[500  1000]; % nr. de simulari
p=0.8; % fiabilitatea componentelor
m=3; % nr. de componente paralel
for j=1:length(n)
k=0;
      for i = 1:n(j)
            x=rand(1,m);
         if any(x<=p)
            k=k+1;
         else
         end
      end
F(1,j)=k/n(1,j) % fiabilitatea
end
```

Se obţin fiabilităţile [0.9920 0.9960].

Programul aferent pct. 3, pentru 5000 de simulări este:

```
n=5000; p1=0.8; p2=0.92;
F1=rand(n,1); F2=rand(n,1);
F=find(F1<=p1 & F2<=p2);
N=length(F); %numarul de functionari
fprintf('Fiabilitatea sistemului este %3.2f\n',N/n)
```

Fiabilitatea sistemului este 0.74.

31.8. Evaluarea prin interpolare a temperaturii

Se consideră o suprafaţă şi o reţea asociată având pas constant. În nodurile reţelei s-au măsurat temperaturile de mai jos:

	1	2	3	4	5	6	7	8
1	14	15	15.5	15.8	16	15	14	13.8
2	14.2	15.4	15.6	15.9	16.5	15.5	15.1	14.3
3	14.5	15.8	15.1	15.3	16.2	15.2	15.3	14.1
4	14.1	15.3	15.2	15.1	16.1	15.4	15.6	14.3
5	14.4	15.6	15.4	15.4	15.8	15.8	15.9	14.8
6	14.8	15.2	14.9	15.3	15.6	15.9	15.4	15.1

Să se evalueze prin interpolare valorile maximă şi minimă ale temperaturii şi să se precizeze cu o rezoluţie de 2% din lungimea pasului reţelei coordonatele punctului în care temperatura are valoarea egală cu media aritmetică a celor două valori extreme.

Condiţia de rezoluţie impusă se obţine pentru un factor de întreţesere a reţelei de măsurare k=6. În tabelul de mai jos se dau valorile rezoluţiei după fiecare nivel de înjumătăţire a pasului reţelei.

Nivel k	0	1	2	3	4	5	6	7
Rezoluţie	100	50	25	12.5	6.25	3.12	1.56	0.78

Cu secvenţa MATLAB:

```
A=[14,15,15.5,15.8,16,15,14,13.8
14.2,15.4,15.6,15.9,16.5,15.5,15.1,14.3
14.5,15.8,15.1,15.3,16.2,15.2,15.3,14.1
14.1,15.3,15.2,15.1,16.1,15.4,15.6,14.3
14.4,15.6,15.4,15.4,15.8,15.8,15.9,14.8
14.8,15.2,14.9,15.3,15.6,15.9,15.4,15.1];
Ai=interp5(A,6);
M=max(max(Ai)); m=min(min(Ai));
a=['Temperatura maxima =', num2str(M) ]
eval(a)
b=['Temperatura minima =', num2str(m) ]
eval(b)
[l,c]=size(A); x=0:l-1; x= x'; y=0:c-1;
```

```
[li, ci]=size(Ai);
xi=0:(l-1)/(li-1):l-1; xi= xi';
yi=0:(c-1)/(ci-1):c-1;
mesh(y,x,A-5); hold on
mesh(yi, xi,Ai); grid
```

se obţine rezultatul:

Temperatura maximă = 16.5060 °C
Temperatura minimă = 13.7750 °C

şi reprezentarea grafică din figura 31.9.

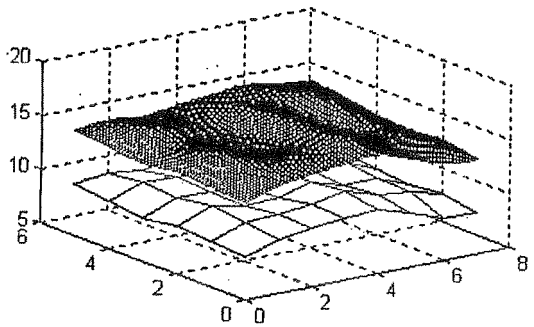


Fig. 31.9. Reprezentarea grafică comparativă a datelor înainte şi după interpolare.

31.9. Temperatura de echilibru a unei plăci

Să se determine distribuţia temperaturii într-o placă subţire aflată la echilibru termic, atunci când se cunosc valorile temperaturii pe cele patru laturi ale acesteia. Trei laturi ale conturului, AB, BC şi CD, figura 31.10, sunt menţinute la aceeaşi temperatură (T_a), în timp ce a patra, DA, este menţinută la o altă temperatură (T_b).

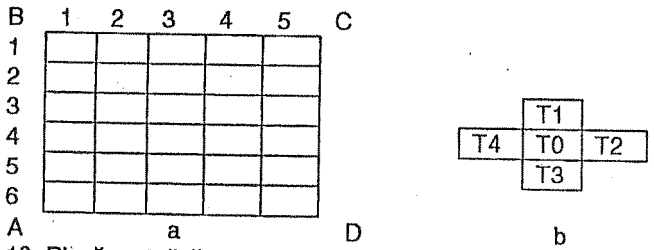


Fig. 31.10. Placă metalică omogenă. a. Reţea de divizare uniformă a plăcii. b.Notaţia temperaturii în puncte vecine.

Este cunoscut faptul că în regim termic staţionar (adică la echilibru termic) câmpul termic satisface ecuaţia Laplace. Considerând o reţea uniformă de divizare în pătrate a suprafeţei plăcii, soluţia ecuaţiei Laplace este aceea pentru care temperatura din orice punct interior al acesteia este media aritmetică a temperaturilor celor patru

puncte vecine, adică:

$$T_0 = \frac{T_1 + T_2 + T_3 + T_4}{4}$$

Se aplică un algoritm de calcul iterativ, în care valorile necunoscute ale temperaturii se setează iniţial cu zero.

După evaluarea noii temperaturi într-un punct oarecare din interiorul domeniului de calcul, se calculează diferenţa dintre această temperatură şi vechea valoare. Dacă eroarea maximă pe întreaga reţea este mai mare decât o valoare specificată, procesul se repetă.

Se utilizează două matrice în care sunt înregistrate temperaturile în nodurile reţelei de calcul la două iteraţii succesive. Sunt necesare două matrice, deoarece temperaturile se reactualizează simultan în toate punctele.

Dacă eroarea rezultatelor a două iteraţii succesive în toate punctele reţelei de calcul este inferioară celei impuse, procesul iterativ se stopează, obţinându-se soluţia aplicaţiei.

Fie o placă plană cu o reţea 6x5 puncte. Temperatura pe trei dintre laturi este 100°C, iar pe a patra este 50°C. Eroarea absolută admisă drept criteriu final al calculului iterativ este de 1°C. Matricea iniţială a temperaturilor şi rezultatele unor iteraţii, inclusiv soluţia aplicaţiei sunt:

Temperatura iniţială

100	100	100	100	100
100	0	0	0	100
100	0	0	0	100
100	0	0	0	100
100	0	0	0	100
50	50	50	50	50

Temperatura după prima iteraţie.

100	100	100	100	100
100	50.0000	25.0000	50.0000	100
100	25.0000	0	25.0000	100
100	25.0000	0	25.0000	100
100	37.5000	12.5000	37.5000	100
50	50	50	50	50

.....

Iteraţia a 13-a.

100	100	100	100	100
100	96.8100	95.5029	96.8100	100
100	93.1795	90.5184	93.1795	100
100	87.8173	83.5133	87.8173	100
100	76.9121	71.3238	76.9121	100
50	50	50	50	50

Soluţia (iteraţia a 14-a)

100	100	100	100	100
100	97.1706	96.0346	97.1706	100
100	93.7864	91.3438	93.7864	100
100	88.4012	84.3692	88.4012	100
100	77.2853	71.8344	77.2853	100
50	50	50	50	50

Programul MATLAB asociat problemei este următorul:

```
%Program pentru calculul temp.de echilibru
nl=input('Introduceti numarul de linii:');
nc=input('Introduceti numarul de coloane:');
t123=input('Introduceti temp. lat.1, 2, 3 :');
t4=input('Introduceti temp. lat. 4:');
tol= input('Introduceti tol. la echilibru:');
tv=zeros(nl, nc);
tv(1,:)= t123+zeros(1, nc);
tv(:, 1) = t123 + zeros(nl, 1);
tv(:, nc) = t123+zeros(nl, 1);
tv (nl, :) = t4 + zeros (1, nc);
disp ('Temperaturile initiale')
disp (tv)
tn=tv;
echilibru = 0
%Actualizarea temp. si testarea echilibrului:
while ~echilibru
for m = 2:nl-1
for n = 2:nc-1
tn(m,n)=(tv(m-1,n)+tv(m,n-1)+tv(m,n+1)+tv(m+1,n))/4
end
end
if all((tn-tv) <= tol)
echilibru = 1
disp('Temperaturile de echilibru')
disp(tn)
end
tv = tn;
end
mesh(tv)
```


31.10. Pătrunderea căldurii în semispaţiul omogen

Câmpul termic nestaţionar $\theta(x,t)$ în semispaţiul x>0, fără surse termice, figura 31.11, caracterizat prin proprietăţile de conducţie termică: conductivitate termică λ, densitatea ρ şi căldura specifică c, satisface ecuaţia lui Fourier:

$$\frac{\rho c}{\lambda}\frac{\partial\theta}{\partial t}=\frac{1}{\alpha}\frac{\partial\theta}{\partial t}=\frac{\partial^2\theta}{\partial x^2}$$

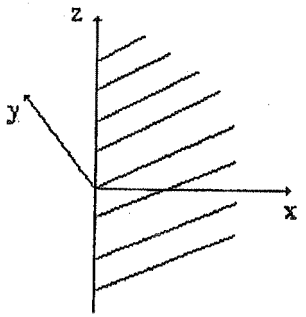


Fig. 31.11

Cu următoarele condiţii iniţială şi la limite:

$\theta(x,0)=0$, pentru orice x>0;
$\theta(0,t)=\theta_0$, pentru orice t>0;
$\theta(\infty,t)=$finit, pentru orice t>0.

soluţia ecuaţiei este:

$$\theta(x,t)=\theta_0\,\mathrm{erfc}\left\{\frac{x}{2\sqrt{\alpha t}}\right\}$$

unde erfc(z) este funcţia complementară a erorilor.

Fluxul termic specific la nivelul suprafeţei x=0 are expresia:

$$p_s=\frac{\lambda\theta_0}{\sqrt{\pi\alpha t}}$$

iar căldura care pătrunde în semispaţiu, raportată la unitatea de suprafaţă, în intervalul de timp (0, t) se calculează cu expresia:

$$Q(t)=\int_0^t p_s(t)dt=\frac{\lambda\theta_0}{\sqrt{\pi\alpha}}2\sqrt{t}$$

Să se scrie un program pentru evaluarea:

1. funcţiei $\theta(x)$ la momentele de timp $t_1=1$, $t_2=5$, $t_3=20$ şi $t_4=60$ secunde ;
2. funcţiei $\theta(t)$ la coordonatele $x_1=1$ mm , $x_2=5$ mm şi $x_3=20$ mm ;
3. fluxului termic specific, $p_s(t)$;
4. căldurii difuzată în semispaţiu, la x=0, Q(t),

cu datele numerice: λ=393,4 W/m°C, ρ=8930 kg/m^3 , c=388,8 J/kg °C şi temperatura suprafeţei θ_0=400 °C.

Secvenţa de program este:

```
clear; clc; clf;
lamda=393.4;   ro=8930;   c=388.8; theta0=400;
thetat0=0;  % conditia  initiala
thetax0=theta0;  % conditia la limita
alfa=lamda/(ro*c);
x=(0:.001:.05)';
t=[1  5  20  60] ;
n=length(t);
subplot(221)
for i=1:n
y1=['theta',num2str(i)];
y2=['=theta0*erfc(x./(2.*sqrt(alfa.*t(',num2str(i),'))));'];
y=[y1 y2];
eval(y)
z=['plot(x,theta',num2str(i), ') '];
eval(z)
hold on
end
axis([0 .05 0 400]); grid
xlabel('Coordonata x'); ylabel('Temperatura ')
title('Variatia spatiala a temperaturii ')
t=(0.0001:.1: 60)';
x=[.001  .005  .020 ] ;
n=length(x);
subplot(222)
for i=1:n
y1=['theta',num2str(i)];
y2=['=theta0*erfc(x(',num2str(i),')./(2.*sqrt(alfa.*t)));'];
y=[y1 y2];
eval(y)
z=['plot(t,theta',num2str(i), ') '];
eval(z)
hold on
end
axis([0 60  0 400]); grid
xlabel('timp '); ylabel('Temperatura ')
title(' Variatia temperaturii in timp')
t=(1:.1: 60)';
ps=lamda*theta0./sqrt(pi*alfa.*t);
ps=ps/10^6;
subplot(223); plot(t,ps); grid
xlabel('timp '); ylabel('Fluxul specific  [x10^6] ')
title('  Fluxul termic specific ')
```



```
Q=lamda*theta0./sqrt(pi.*alfa)*2*sqrt(t);
Q=Q/10^8;
subplot(224);  plot(t,Q); grid
xlabel('timp '); ylabel('Caldura [x10^8] ')
title('  Caldura difuzata in semispatiu ')
```

Rezultatele grafice sunt prezentate în figura 31.12.

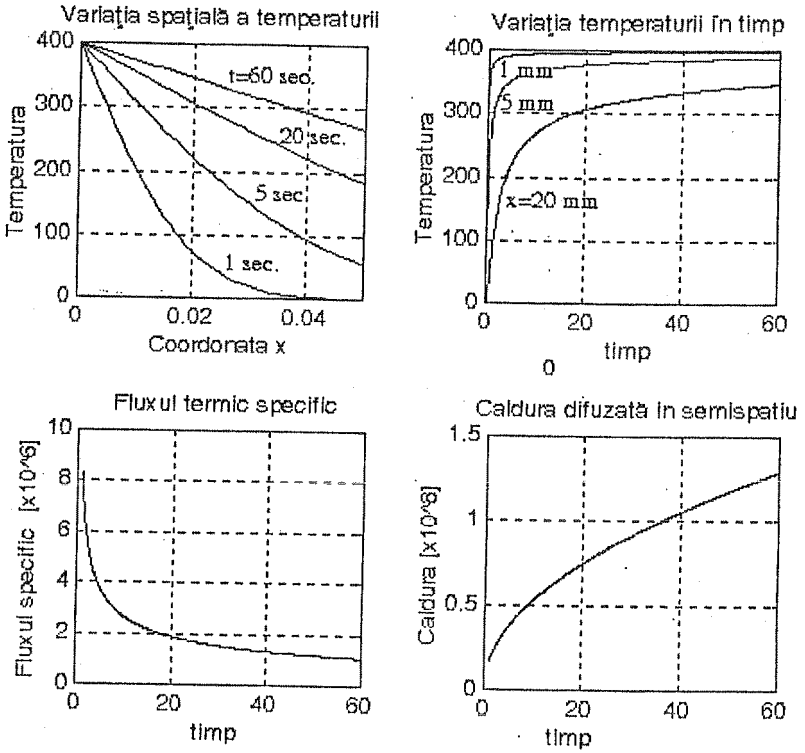


Fig. 31.12.

31.11. Timpul de topire al unui fir fuzibil

Un fir fuzibil, figura 31.13, cu diametrul d şi lungimea l, are în momentul iniţial o temperatură care variază liniar între valorile θ_A şi θ_B la cele două extremităţi, moment în care este supus unui curent electric de intensitate I. Firul este izolat termic, iar în procesul tranzitoriu de încălzire extremităţile rămân la temperatura iniţială. Temperatura de topire a firului este θ_t .

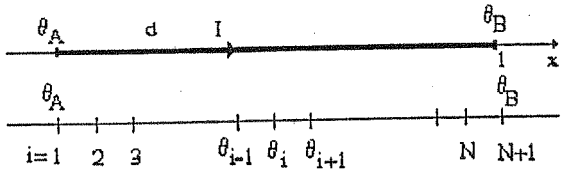


Fig.31.13.

Să se scrie un program de studiu al regimului termic tranzitoriu şi să se determine timpul de topire al firului pentru datele numerice: γ =8900 kg/m^3, c=400J/kg°C , ρ =0.025· 10^{-6} Ωm, λ=390 W/m°C, d=0.3 mm, I=100 A, l=0.04 m, θ_A=20°C, θ_B=100°C, θ_t=1090°C.

Temperatura tranzitorie $\theta(x,t)$ a firului satisface ecuaţia:

$$\frac{\gamma c}{\lambda}\frac{\partial\theta}{\partial t}=\frac{1}{\alpha}\frac{\partial\theta}{\partial t}=\frac{p}{\lambda}+\frac{\partial^2\theta}{\partial x^2}$$

unde γ, c şi λ sunt densitatea, căldura specifică şi conductivitatea termică ale materialului firului, iar p este densitatea volumică a puterii de efect Joule:

$$p=\frac{16\rho I^2}{\pi^2 d^4}$$

unde ρ este rezistivitatea materialului firului. Funcţia $\theta(x,t)$ satisface condiţia iniţială:

$\theta(x,0)=0$, pentru orice 0<x<l;

şi condiţiile la limite:

$\theta(0,t)=\theta_A$, $\theta(l,t)=\theta_B$, pentru orice t>0;

Forma în diferenţe finite a ecuaţiei pentru o divizare în N intervale echidistante, fig. 31.12, este:

$$\frac{1}{\alpha}\frac{\theta_i^{(k+1)}-\theta_i^{(k)}}{\Delta t}=\frac{p}{\lambda}+\frac{\theta_{i-1}^{(k)}-2\theta_i^{(k)}+\theta_{i+1}^{k}}{h^2}$$

unde $\theta_{i-1}^{(k)}$, $\theta_i^{(k)}$, θ_{i+1}^{k} sunt temperaturile în nodurile i-1, i şi i+1 la momentul de timp t_k, $\theta_i^{(k+1)}$ este temperatura în nodul i la momentul de timp $t_{k+1}=t_k+\Delta t$. Pasul reţelei de divizare a firului este h=l/N.

Cu notaţiile $\beta=\frac{\lambda}{\gamma c}\frac{\Delta t}{h^2}$ şi $\delta=\frac{\Delta t\cdot p}{\gamma c}$, ultima relaţie se rescrie sub forma:

$$\theta_i^{(k+1)}=\delta+\beta\theta_{i-1}^{(k)}+(1-2\beta)\theta_i^{(k)}+\beta\theta_{i+1}^{(k)}$$

Această relaţie permite determinarea valorilor temperaturii în toate nodurile i=2, 3, ..., N, la momentul de timp t_{k+1}, atunci când sunt cunoscute temperaturile la momentul de timp anterior t_k.

Algoritmul de integrare numerică prin schema cu diferenţe finite este următorul:

1. - se alege numărul N al paşilor de discretizare a firului şi pasul de timp Δt;
2. - se calculează valorile $\theta_i^{(1)}$ ale temperaturii la primul pas de timp $t_1=\Delta t$, pe baza câmpului termic iniţial;

3. - se calculează apoi succesiv valorile $\theta_i^{(k+1)}$ ale temperaturii, k=1, 2, 3..., la momentele de timp $t_2=2\Delta t$, $t_3=3\Delta t$, ...;
4. - timpul de topire este acela pentru care temperatura maximă în lungul firului a atins valoarea θ_t.

Observaţie: Funcţionarea corectă a programului depinde de fineţea divizării firului şi de corelarea mărimii pasului de timp cu intensitatea curentului.

Cu programul:

```
clear; clc;
lamda=390; ro=0.025*10^(-6);
d=0.3*10^(-3); gama=8900;
c=400; I=100; l=0.04;
TA=20;  TB=100;  Tt=1090; % temp. initiala si de topire
A=lamda/(gama*c);
p=16*ro*I^2/(pi^2*d^4);
N=50;  %nr. de diviziuni ale firului
dt=0.001; % pasul de timp
h=l/N;
B=lamda/(gama*c)*dt/h^2;
D=p*dt/(gama*c);
Ta(1:N+1)= TA+(TB-TA).*(0:N)'/N;
Tmax=TA
timp=0;
while (Tmax-Tt)<=0
timp=timp+dt;
Ta(1)=TA;
Ta(N+1)=TB;
Ta(2:N)=D+B*Ta(1:N-1)+(1-2*B)*Ta(2:N)+B*Ta(3:N+1);
Tmax=max(Ta)
end
timp
plot(Ta)
```

se obţine timpul de topire al fuzibilului 0.2940 secunde.

31.12. Câmpul magnetic armonic într-un conductor cilindric

Un conductor cilindric circular infinit lung de rază a este situat într-un câmp magnetic axial, variabil în timp, de forma:

$$H=\sqrt{2}H_0\sin(\omega t)$$

Prin integrarea ecuaţiei Helmoltz:

$$\Delta\underline{H}=\frac{d^2\underline{H}}{dr^2}+\frac{1}{r}\frac{d\underline{H}}{dr}=\gamma^2\underline{H}$$

unde $\underline{H}$ este imaginea în complex a intensităţilor câmpului magnetic, cu condiţiile la limită:

$$\underline{H}(a)=H_0 \quad \text{şi} \quad \underline{H}(0)=\text{finit}$$

se obţine funcţia de variaţie a intensităţii câmpului în interiorul cilindrului:

$$\underline{H}(r) = H0\frac{I_0(\underline{\gamma}\, r)}{I_0(\underline{\gamma}\, a)}$$

unde I_0 este funcţia Bessel de ordin zero, iar mărimea complexă $\underline{\gamma}$ are expresia $\underline{\gamma} = (1+j)\sqrt{\frac{\omega\mu\gamma}{2}}$. Variaţia radială a câmpului este dependentă de frecvenţa $f = \frac{\omega}{2\pi}$ a câmpului şi de proprietăţile de material: permeabilitatea magnetică μ şi conductivitatea electrică $\sigma=1/\rho$ (ρ este rezistivitatea).

Puterea activă iundusă pe unitatea de lungime a cilindrului are expresia:

$$P = 2\pi a\sqrt{\pi f\mu\rho}H_0^2\,\mathrm{Real}\left\{(1+j)\frac{I_1(\underline{\gamma}\, a)}{I_0(\underline{\gamma}\, a)}\right\}$$

unde I_1 este funcţia Bessel de ordinul 1.

Să se scrie un program pentru calculul puterii induse şi pentru reprezentarea grafică a variaţiei radiale $H(r)/H_0 = |I_0(\underline{\gamma}\, r)|/|I_0(\underline{\gamma}\, a)|$ a câmpului în interiorul cilindrului.

Pentru aplicaţia numerică se consideră următoarele date: a=50 mm; ρ=1 Ωmm^2/m; $\mu = \mu_0 = 4\pi\cdot10^{-7}$ H/m; H_0=2000 A/cm; f=1000 Hz, 2500 Hz, 8000 Hz.

Cu programul:

```
a=0.05;  ro=10^(-6);
miu=4*pi*10^(-7);
H0=2*10^5;
frecv=[1000  2500  8000];
sigma=1/ro;
for k=1:3
f=frecv(k);
omega=2*pi*f;
gama=(1+j)*sqrt((omega*miu*sigma)/2);
I0a=bessel(0,gama*a);
I1a=bessel(1,gama*a);
P=2*pi*a*sqrt(pi*f*miu*ro)*H0^2*abs(real((1+j)*I1a./I0a));
A=['Frecventa = ' ,num2str(f), ' [Hz]'];
B=['Puterea = ', num2str(P), ' [W]'];
C=[A '          ' B];
disp(C)
r=linspace(0, a);
I0r=bessel(0,gama*r);
Hr0=abs(I0r)/abs(I0a);
hold on
plot(r,Hr0)
end
```



```
axis([0 a 0 1])
grid
xlabel('Coordonata r [m]');
ylabel('H(r)/H0')
title('Variatia radiala a campului magnetic')
```

se obţin rezultatele numerice:

Frecvenţa = 1000 [Hz] Puterea = 6.625e+005 [W]
Frecvenţa = 2500 [Hz] Puterea = 1.12e+006 [W]
Frecvenţa = 8000 [Hz] Puterea = 2.106e+006 [W]

şi variaţia rdială a intensităţii câmpului magnetic pentru cele trei frecvenţe din figura 31.14.

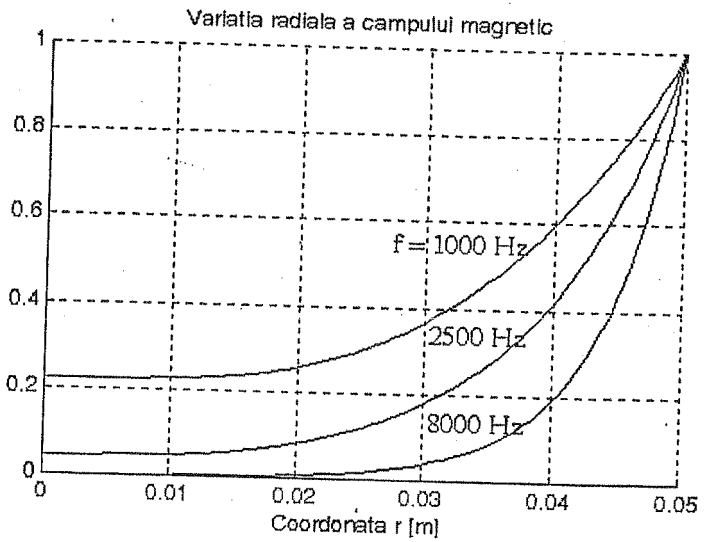


Fig. 31.14.

31.13. Interfaţă grafică interactivă

Să se scrie un program pentru crearea unei interfeţe grafice interactive, cu opţiunile:

- controlul culorilor (Colormap) în meniul principal;
- preluare de semnal din fişier (ACHIZIŢIE);
- reprezentări în diverse descrieri (Semnal temporal; Spectrul de amplitudini; Spectrul de putere; Spectrograma; Transformata Wigner-Ville);
- buton de închidere a interfeţei (STOP)

Programul care realizează cerinţele impuse prin enunţ este:

```
function interf2(action);
%Program pentru interfata grafica interactiva
if nargin<1,
     action='start';
end;
%coordonatele ferestrei grafice si a axelor pe ecran
ps=0.01;     %distanta ferestrei fata de stinga
pj=0.3;      %distanta ferestrei fata de jos
lg=0.98;     % lungimea ferestrei
hg=0.65;     %inaltimea ferestrei
global DISP
DISP=1;
S = brighten([[zeros(8,2) (3:10)'/10]; prism(56)],1/3);
if strcmp(action,'start'),
    clf reset;
    set(gcf,'Units','normalized','backingstore','off','color',...
         'blue','NumberTitle','off','colormap',S,...
         'Position',[.01 .25 .98 .65],...
         'Name','ANALIZOR  SPECTRAL  de  INALTA  REZOLUTIE');
    colormenu;
%coordonatele butoanelor de pe ecran
n_semn=1;              % numarul de tipuri de semnale
n_buts = n_semn+1; % numarul de butoane semnal + inchidere
n_butr=4;              % numarul de butoane pentru reprezentari
n_but=n_buts+n_butr+3; % numarul de butoane
echivalente(+spatiile)
% se plaseaza butoanele meniului in partea stinga
h_gl=1-(n_but+1)*.01; %inaltimea echivalenta pentru butoane
h_but =.9/n_but;  %inaltimea butonului
l_but = 0.27;     % lungimea butonului
d_s1=.01;         % distanta fata de marginea din stinga
d_j1=.01;         % distanta fata de marginea de jos
% se pune butonul de inchidere 'STOP'
pos = [d_s1   d_j1*.5   l_but   h_but];%pozitia  butonului
hcl = uicontrol(gcf,'units','normal','pos',pos,...
         'String','STOP','Callback','interf2(''close'')');
% se pune butonul pentru semnalele achizitionate
%(meniul din stinga)
pos=[d_s1  (h_but*(1+n_butr+3)+d_j1)+.025  l_but  h_but-.02];
global ACHIZ
ACHIZ=uicontrol(gcf,'Units','Normal','Position',pos,...
'String','ACHIZITIE','Callback','interf2(''achizitie'')');
% se pun butoanele radio pentru functiile de afisat
b_st=uicontrol(gcf,'Style','RadioButton','Units','Normal',...
       'Position',[d_s1  (h_but*6+d_j1)  l_but  .95*h_but],...
       'ForegroundColor','blue','String','Semnal temporal',...
```



```
       'Callback','interf2(''temp'')');
b_spa=uicontrol(gcf,'Style','RadioButton','Units','Normal',...
       'ForegroundColor','blue', 'Position',...
       [d_s1  (h_but*5+d_j1)  l_but  .95*h_but],'String',...
       'Spectrul de amplitudine','Callback',...
       'interf2(''amplitd'')');
b_spp=uicontrol(gcf,'Style','RadioButton','Units','Normal',...
       'ForegroundColor','blue','Position',...
       [d_s1 (h_but*4+d_j1) l_but  .95*h_but],'String',...
       'Spectrul de putere','Callback','interf2(''putere'')');
b_spg=uicontrol(gcf,'Style','RadioButton','Units','Normal',...
       'ForegroundColor','blue','Position',...
       [d_s1 (h_but*3+d_j1) l_but .95*h_but],'String',...
       'Spectrograma','Callback','interf2(''spectrogr'')');
b_wv=uicontrol(gcf,'Style','RadioButton','Units','Normal',...
       'ForegroundColor','blue','Position',...
       [d_s1 (h_but*2+d_j1)  l_but  .95*h_but],...
        'String','Wigner-Ville','Callback','interf2(''w-v'')');
%se pozitioneaza axele in fereastra grafica
global SD_AXIS
SD_AXIS = axes('position',[1.35*l_but .15 (1-1.5*l_but) .75]);
axis off
set(SD_AXIS,'position',[1.35*l_but .15 (1-1.5*l_but) .75],...
       'XColor',[1 0 0],'YColor',[1 0 0],'ZColor',[1 0 0])
get(SD_AXIS)
%se pune titlul meniului 'semnale'
d_s=.05;        %deplasarea supratitlurilor fata de stinga
rect=[d_s+.04  ((n_but)*h_but+d_j1)  (l_but-d_s)  h_but-.04];
uicontrol(gcf,'style','text','units','normal','position',rect,...
      'string','SEMNALE','backgroundcolor',get(gcf,'color'),...
      'foregroundcolor','red','horizontal','left');
%  se pune titlul functiilor 'Reprezentarile'
rect = [.05 .715 .19 .08181] ;
uicontrol(gcf,'style','text','units','normal','position',rect,...
    'string','REPREZENTARI','backgroundcolor',get(gcf,'color'),...
     'foregroundcolor', 'red','Horizontal','left');
%se creeaza semnalul initial
fe=1000;
t=0:.001:.12;
s=1.5*sin(2*pi*100*t);
N=length(N);
set(gcf,'UserData',[t(:) s(:)]);
global DT; global DF; global PCT
DT=[t(:) ;s(:)];
DF=[0 ;0 ;0];
PCT=[N ;0];
```

```
elseif strcmp(action,'achizitie'),
clear
   t_0=clock;
   mesaj=uicontrol('Style','Edit','Position',[.48 .45 .25 .1],...
              'Units','Normalized','BackgroundColor','Black',...
              'ForegroundColor','green',...
              'String','ASTEPTATI,VA ROG...',...
              'HorizontalAlignment','Center','Clipping','on');
      [nf,cf] = uigetfile('*.asc', 'Load');
      x_=['load ',nf];
      eval(x_);
      n_=max(size(nf));
      for i=1:n_
          if nf(1,i)=='.'
             i;
             break
          end;
      end;
      numeX=nf(1,1:(i-1));
      extX=nf(1,(i+1):n_);
      x_=['x=',numeX,';'];
      eval(x_);
      t=x(:,1);
      s=x(:,2);
      N=length(t);
      DT(1:N)=t;
      DT(N+1:N+N)=s;
      PCT(1)=N;
      delete(mesaj);
      set(gcf,'UserData',[t(:) s(:)]);
      interf2('temp');
elseif strcmp(action,'temp'),
      DISP=1;
      y=get(gcf,'UserData');
      if ~isempty(y),
           t=y(:,1);
           s=y(:,2);
      end;
      plot(t,s,'r-');grid;title('SEMNAL ACHIZITIONAT');
elseif strcmp(action,'amplitd'),
     DISP=3;
     y=get(gcf,'UserData');
     if ~isempty(y),
          t=y(:,1);
          s=y(:,2);
     end;
```



```
     Nf=max(size(s));
     s1=boxcar(Nf).*s;
     F=fft(s1);
     M=length(F);
     Fm=abs(F);
     Fm(1,1)=Fm(1,1)/M;
     Fm(2:M,1)=Fm(2:M,1)/(M/2);
     F=Fm(1:M/2+1,1);
     fe=1/(t(2)-t(1));
     fw=(0:M/2)*fe/M;
     semilogy(fw',F,'r-');
     title('SPECTRUL de AMPLITUDINE');
     xlabel('Frecventa  [Hz]');
     PCT(2)=M;
     DF(1:M/2+1)=fw;
     DF(M/2+1+1:M/2+1+M/2+1)=F;
     axis([0 fe/2 .0005 20]);
elseif strcmp(action,'spectrogr'),
     DISP=2;
     y=get(gcf,'UserData');
     if ~isempty(y),
          t=y(:,1);
          s=y(:,2);
     end;
     specgram(s);
     axis on;xlabel('timp [sec]');
     ylabel('frecventa [Hz]');
elseif strcmp(action,'putere'),
     y=get(gcf,'UserData');
     if ~isempty(y),
          t=y(:,1);
          s=y(:,2);
     end;
     N=max(size(s));
     Nf=200;
     c=zeros(Nf,N);
     s1=[zeros(Nf/2,1);s;zeros(Nf/2,1)];
     d=c;
     for i=1:N,
       c(:,i)=s1(i:i-1+Nf,1);
       d(:,i)=triang(Nf).*c(:,i);
     end
     D=fft(d);
     clear d
     clear c
     [dl dc]=size(D);
```

```
      D1=abs(D(1:dl/2+1,:));
      D1=D1(:,:)/(Nf/16);
      clear D
      fe=1/(t(2)-t(1));        % frecventa de esantionare
      fd=0:fe/Nf:fe/2;
      plot(fd,D1);grid
      xlabel('Frecventa     [Hz]');
      title('SPECTRUL de PUTERE');
elseif strcmp(action,'w-v'),
      y=get(gcf,'UserData');
      if ~isempty(y),
            t=y(:,1);
            s=y(:,2);
      end;
      N=max(size(s));
      Nf=200;
      c=zeros(Nf,N);
      s1=[zeros(Nf/2,1);s;zeros(Nf/2,1)];
      d=c;
      for i=1:N,
        c(:,i)=s1(i:i-1+Nf,1);
        d(:,i)=hamming(Nf).*c(:,i);
      end
      D=fft(d);
      clear d
      clear c
      [dl dc]=size(D);
      D1=abs(D(1:dl/2+1,:));
      clear D
      fe=1/(t(2)-t(1));     % frecventa de esantionare
      fd=0:fe/Nf:fe/2;
      D2=D1(:,1:8:N);
      ss=s(1:8:N);
      mesh(ss,fd,D2),surface(D2);surfl(D2);
       set(gca,'box','on');grid;
       title('SPECTRUL de PUTERE');
       xlabel('TIMPUL [ms]');ylabel('FRECVENTA [Hz]');
elseif strcmp(action,'close'),
      y=get(gcf,'UserData');
      clf reset;
      clear ACHIZ;
      clear SD_AXIS;
      clear DISP;
      close;
end;
```


Cu programul de mai sus se obţine interfaţa grafică interactivă din figura 31.14.

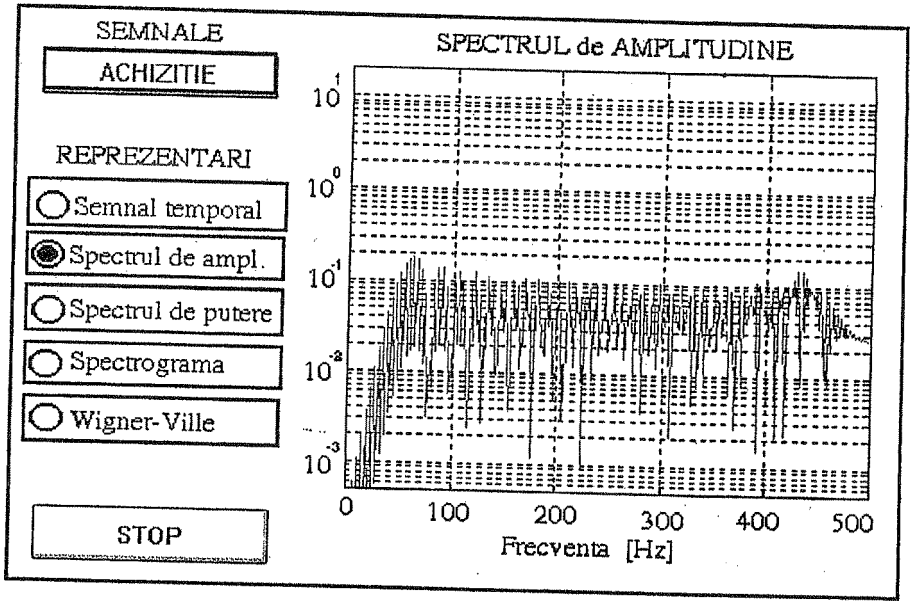


Fig. 31.15. Interfaţă grafică interactivă.

ANEXĂ
Indexul funcţiilor MATLAB

abs 63, 93, 94
acos 100
acosh 101
acot 100
acoth 101
acsc 100
acsch 101
all 58
and 49
angle 93, 94
ans 35
any 58
asec100
asech 101
asin 100
asinh 101
atan 100
atan2 100
atanh 101
axes 216
axis 216, 217

bar 136, 192, 193
bessel 182, 183
besseli 182, 183
besselj 182, 183
besselk 182, 183
bessely 182, 183
beta 182, 185
betainc 182, 185
betaln 182, 185
blanks 63, 65
bone 251, 252
break 47, 57
brighten 243, 248

caxis 243, 245
cd 30, 31
ceil 89
chol 123, 125
cla 216, 218
clabel 201, 203
clc 32, 33
clear 34
clf 225, 230
clock 32
close 225, 230
colormap 243, 244
ColorSpec 243
comet 193, 200
comet3 201, 207
compan 102
compass 192, 198
computer 36
cond 118, 122
condest 118, 122
conj 93, 96
contour 201, 202
contour 201, 203
contourc 201
contrast 243
conv 143, 145
cool 251, 252
copper 251, 252
corrcoef 129, 134
cos 99
cosh 101
cot 100
coth 101
cov 129
cputime 32
csc 100
csch 101
cumprod 129, 131
cumsum 129, 131
cylinder 210

date 32
deblank 63, 65
dec2hex 69
deconv 143, 145
del2 171
delete 30, 239
det 118
diag 102, 103, 112, 117
diary 30, 31
diff 129, 168
diffuse 249, 251
dir 30, 31
disp 34
drawnow 239, 242

echo 32, 33
eig 123
else 47, 53
elseif 47, 53
end 47, 53, 55
eps 35
erf 182, 187
erfc 182, 187
erfinv 182, 187
error 47, 58
errorbar 192, 195
etime 32
eval 63, 65
exist 58, 61
exit 27
exp 96, 99
eye 82, 86

feather 192, 198
feval 63, 66
figure 225
fill 198, 191
fill3 201, 206
find 58, 59, 135
findstr 66, 67
finite 58, 60
fix 89
flag 251, 252

fliplr 112, 116
flipud 112, 116
floor 89
flops 36
fmin 137
fmins 137, 138
for 47, 55
format 32, 33
fplot 193, 199
function 42
fzero 137, 139

gamma 182, 185
gammainc 182, 185
gammaln 182, 185
gca 216, 218
gcd 93
gcf 225
get 239, 240
getframe 255
ginput 259, 266
gradient 172
gray 251, 253
grid 214, 215
griddata 149, 158
gtext 214, 215

hadamard 102, 104
hankel 102, 105
help 27, 28, 44
hex2dec 69
hidden 212, 213
hilb 102, 106
hist 129, 136, 192, 196
hold 216, 218
home 32, 33
hot 251, 253
hsv 251, 253
hsv2rgb 243, 247

i 36
if 47, 51
imag 93, 96
image 231, 237
Inf 36
int2str 67, 68
interp1 149, 156
interp2 149, 159
interp3 149, 156
interp4 149, 156
interp5 149, 157
interpft 149, 154
inv 118, 119
invhilb 102, 106
ishold 216
isieee 36
isinf 58, 60
isletter 66, 67
isnan 58, 60, 135
isstr 63, 64

j 36
jet 251, 254

kron 102, 107

lcm 93
length 34, 35
line 231, 233
linspace 82
load 36
log 96, 98
log10 96, 98
log2 96, 98
loglog 189, 191
logspace 82, 83
lookfor 27, 28
lower 66, 67
lu 123, 126

magic 102, 107
matlabrc 27
max 129
mean 129, 130
median 129, 130
mesh 207, 208
meshc 207, 208
meshgrid 82, 87
meshz 207, 208
min 129
more 32
movie 255
moviein 255

NaN 36
nargin 36
nargout 36
newplot 239, 240
nextpow2 96, 97
norm 118, 120
not 49
num2str 67, 68

ode23 175
ode45 175
ones 82, 84
or 49
orient 257, 258

pack 34, 35
pascal 102, 108
path 27, 29
pcolor 243, 246
pi 35
pink 251, 254
pinv 123, 127
plot 189
plot3 201, 205
polar 192, 193
poly 143, 148
polyder 143, 147
polyfit 149, 160, 161
polyval 143, 144
pow2 96, 97
print 257
printopt 257
prism 251, 254
prod 129, 131

qr 123, 127
quad 166
quad8 166
quit 27
quiver 172, 201, 204
qz 123

rand 82, 86
randn 82, 86
rank 118, 119
rat 89, 91
rats 89, 90
rcond 118, 122
real 93, 96
realmax 36

realmin 36
rem 89, 92
reset 239, 243
reshape 112, 115
residue 143, 146
return 47, 58
rgb2hsv 243, 247
rgbplot 243, 249
roots 143, 148
rose 192, 197
rot90 112, 117
round 89
rref 123

save 36
script 42
sec 100
sech 101
semilogx 189, 191
semilogy 189, 191
set 239
setstr 63, 64
shading 243, 246
sign 89, 92
sin 99
sinh 101
size 34, 35
sort 129, 132
specular 249, 251
sphere 210, 211
spinmap 243, 248
spline 149, 153
sprintf 67, 68
sqrt 96, 98
stairs 192, 195
startup 27
std 129, 134
stem 192, 194
str2mat 63, 64
str2num 67, 68
strcmp 66
strings 63
subplot 216
sum 129, 131
surf 207, 209
surface 231, 235
surfc 207, 209
surfl 249, 250
surfnorm 249, 250
svd 124

table1 149
table2 149, 150
tan 100
tanh 101
text 214, 231
tic 32
title 214
toc 32
toeplitz 102, 109
trace 118, 120
trapz 166, 167
tril 112, 117
triu 112, 117
type 27, 29

uicontrol 259, 260
uigetfile 259, 266
uimenu 259, 264
uiputfile 259, 269
uisetcolor 259, 270
uisetfont 259, 269
unwrap 93, 94
upper 66, 67
vander 102, 110
ver 36
version 36
view 212

what 27, 28
which 27
while 47, 57
who 27, 30
whos 27, 30
wilkinson 102, 111

xlabel 214

ylabel 214

zeros 82, 85
zlabel 214

Operatorii. logici : & | ~ 49
Operatorii aritmetici : + - * / \ ^ ' 71
Operatorii relaţionali: < <= > >= == ~= 47
Caractere speciale : [ ] ( ) = , . ' ; % ! : 30, 31, 82

BIBLIOGRAFIE

1. MATLAB. High-Performance Numeric Computation and Visualisation Software. Reference Guide, The MATH WORKS Inc., Natick, Massachusetts, 1992.
2. SCHARF, L. L., BEHREUS, R. T. - A First Course in Electrical and Computer Engineering with MATLAB Programs and Experiments, Addison-Wesley Publishing Company, Reading, Massachusetts, 1990.
3. ETTER, D. M. - Engineering Problem Solving with MATLAB, Prentince Hall, New Jersey, 1993.
4. JENNINGS, A., McKEOWN, J. J. - Matrix Computation, Wiley and Son, New York, 1992.
5. MARCUS, M. - Matrices and MATLAB: a Tutorial, Prentince Hall, New Jersey, 1993.
6. PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., VETTERLING, W. T. - Numerical Recipes in C. The Art of Scientific Computing, Cambridge University Press, 1992.
7. MOCANU, C. - Teoria cîmpului electromagnetic, Editura Didactică şi Pedagogică, Bucureşti, 1981.
8. POSTOLACHE, M. - Metode numerice, Editura Sirius, Bucureşti, 1994.
9. SIMA, V. - Metode noi de matematică aplicată, Editura Ştiinţifică, Bucureşti, 1992.
10. *** Mică enciclopedie matematică, Editura Tehnică, Bucureşti, 1980.